NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

INTEGRATION OF A MULTI-RATE POSITION FILTER IN THE NAVIGATION SYSTEM OF AN UNMANNED AERIAL VEHICLE (UAV) FOR PRECISE NAVIGATION IN THE LOCAL TANGENT PLANE (LTP)

by

Robert C. Perry

March, 1998

Thesis Advisor: Isaac. I. Kaminer

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1998	Master's Thesis

4. TITLE AND SUBTITLE
INTEGRATION OF A MULTI-RATE POSITION FILTER IN THE NAVIGATION SYSTEM OF AN UNMANNED AERIAL VEHICLE (UAV) FOR PRECISE NAVIGATION IN THE LOCAL TANGENT PLANE (LTP)

5. FUNDING NUMBERS

6. AUTHOR(S)
Perry, Robert C.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Differential Global Positioning System (DGPS) provides highly accurate position information but at update rates of one Hz which is inadequate for precise aircraft terminal maneuvering such as take-off and landing. During this period between updates an accurate position estimate in Local Tangent Plane (LTP) can be made using complementary filtering of the DGPS position and indicated airspeed. Use of indicated airspeed as the filter velocity input necessitates the transformation from body to inertial (LTP) reference frame using Euler angle information available from the Inertial Measuring Unit (IMU) or DGPS. This filter provides accurate estimates of both vehicle position and existing wind. These filter outputs of position and wind can then be used as inputs to a trajectory controller to ultimately enable autonomous launch and recovery of an Unmanned Aerial Vehicle.

14. SUBJECT TERMS
Differential Global Positioning System (DGPS), Unmanned Aerial Vehicle (UAV), Local Tangent Plane (LTP), Inertial Measuring Unit (IMU), Euler Angles, Complementary Filter

15. NUMBER OF PAGES 74

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFI- CATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Approved for public release; distribution is unlimited

INTEGRATION OF A MULTI-RATE POSITION FILTER IN THE NAVIGATION SYSTEM OF AN UNMANNED AERIAL VEHICLE (UAV) FOR PRECISE NAVIGATION IN THE LOCAL TANGENT PLANE (LTP)

Robert C. Perry
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1985

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1998

ABSTRACT

Differential Global Positioning System (DGPS) provides highly accurate position information but at update rates of one Hz which is inadequate for precise aircraft terminal maneuvering such as take-off and landing. During this period between updates an accurate position estimate in Local Tangent Plane (LTP) can be made using complementary filtering of the DGPS position and indicated airspeed. Use of indicated airspeed as the filter velocity input necessitates the transformation from body to inertial (LTP) reference frame using Euler angle information available from the Inertial Measuring Unit (IMU) or DGPS. This filter provides accurate estimates of both vehicle position and existing wind. These filter outputs of position and wind can then be used as inputs to a trajectory controller to ultimately enable autonomous launch and recovery of an Unmanned Aerial Vehicle.

TABLE OF CONTENTS

ACKNOWLEDGMENT

I would like to take this opportunity to thank the many people who made the completion of this project possible. Dr. Isaac Kaminer for his guidance, patience and enthusiasm in the endless pursuit for perfection. Steve Froncillo and John Komlosy without whom the last two years would have been much less like being back in the ready room. Most of all, to the four people who helped the most in their support and understanding of the things they did not understand, my wife Brooke and three wonderful children Max, Mason and Ali. Thanks for everything.

I. INTRODUCTION

Operational use of Unmanned Aerial Vehicles (UAV) continues to expand throughout the U.S. Armed Forces. This has placed increasing demands on the accuracy of the navigation systems used to guide these remotely controlled or autonomous vehicles. Future tasks such as autonomous landings aboard deployed vessels or land-based expeditionary fields require precise navigation. Differential Global Positioning System (DGPS) available today is very accurate but offers relatively slow position update rates of one time a second. Inertial Navigation Systems (INS) provides updates at much higher rates but lacks the accuracy of DGPS due to numerous inherent errors. These two systems have been integrated to provide a precise navigation system.

While DGPS is far more accurate for position information than the INS its slow 1 Hz update rate is a distinct drawback in guidance systems requiring accurate position estimates on demand. The goal of this project was to improve on the accuracy of the navigation system employed on the FOG-R Unmanned Air Vehicle (FROG UAV). This was to be accomplished through the use of a complementary filter to provide position and wind estimates in between DGPS updates. Development of this filter was accomplished using the $Matrix_x$ product family of rapid prototyping software available from Integrated Systems Incorporated (ISI). The development process utilized the Realsim, Xmath, and System Build design tools with their associated Graphical User Interface (GUI) to step through the design process. This control system design and implementation software package was executed on the Texas Instruments TMS320C30 Digital Signal Processor (C30_DSP) hosted on an IBM compatible PC. The GPS system used throughout this

thesis utilizes two Motorola PVT-6 OEM receivers. The IMU used is the Watson IMU-600AD. This thesis describes the development of the position and angle filters for use in the navigation system as integrated with the existing FROG command and control.

The FROG UAV used for this project is a remotely controlled (RC) aircraft with a ten foot wing span and 20 pound payload capacity. The aircraft is equipped with an avionics suite consisting of an Inertial Measuring Unit (IMU), Differential Global Positioning System (DGPS), and air data sensors. It is controlled through the use of a Radio Frequency (RF) link that sends Pulse Width Modulated (PWM) signals to the onboard autopilot. The RF link was modified to accommodate control by a ground station consisting of SPARC workstation, luggable C30 DSP and communications box. The onboard GPS was used in the differential mode with the ground station DGPS placed on a surveyed point at the flight test airfield, sending differential corrections to the UAV.

II. BACKGROUND

To understand the implementation of the position filter some background must be introduced. The following discussion will address the sensors used in the implementation of the filter. It utilizes several different coordinate systems, including Local Tangent Plane (LTP), Body-Fixed and Wind systems. The fundamental relationship between linear position and velocity, which are the basis for the development of this filter, is also addressed.

A. DIFFERENTIAL GLOBAL POSITIONING SYSTEM

The integration of DGPS into the FROG navigation system was accomplished in a previous thesis project. A component level discussion of the DGPS system is not necessary for the purposes this thesis. It is essential, however, to be familiar with the characteristics of the position data provided by the DGPS. While the accuracy of this information in a static environment is typically within three meters, the best case position update rate remains relatively slow at once a second (1 Hz). Considering the spectrum of average speeds of platforms on which DGPS can be employed, the FROG UAV of 80 feet per second (fps) to the modern tactical jet at 500 fps, it is evident that extremely precise flight path and ground track navigation is impossible using only GPS available at that rate. For this reason DGPS has been integrated with the Inertial Navigation System. [Ref. 1]

Vehicle position is provided by the DGPS in the form of latitude, longitude and geoide height. As mentioned earlier, the task of converting that position information for use in the local tangent plane (LTP) has already been accomplished. The filter uses this vehicle position data provided in the LTP frame in units of feet. This data is used as an input to the filter to form the position error signal when compared to the position estimates developed by the filter.

B. INERTIAL MEASURING UNIT

Unlike the DGPS, the Inertial Measuring Unit is a self–contained unit of accelerometers and gyros which measure aircraft specific acceleration, angular rates and inertial orientation. The position filter needs to know the orientation of the aircraft's body axes with respect to the inertial frame. These quantities, known as Euler angles ϕ, θ and ψ, define this orientation in roll, pitch and yaw, respectively. For the purposes of this project it is sufficient to treat the local tangent plane as the inertial or universal frame. One of the tasks of this project was to determine whether the Euler angle information provided by the IMU would be suitable for use in the position filter. Typically these quantities are filtered [Ref. 2] to produce accurate estimates of the angles which are ultimately available for use in the navigation system. Specifics on the suitability of the Euler information provided by the IMU for use in the filter are discussed in Chapter VI.

C. COORDINATE SYSTEMS

1. Local Tangent Plane

The LTP coordinate system is defined by passing a plane through any point on the earth's surface with that plane being tangent to the surface at that point as shown in Figure 2.1.

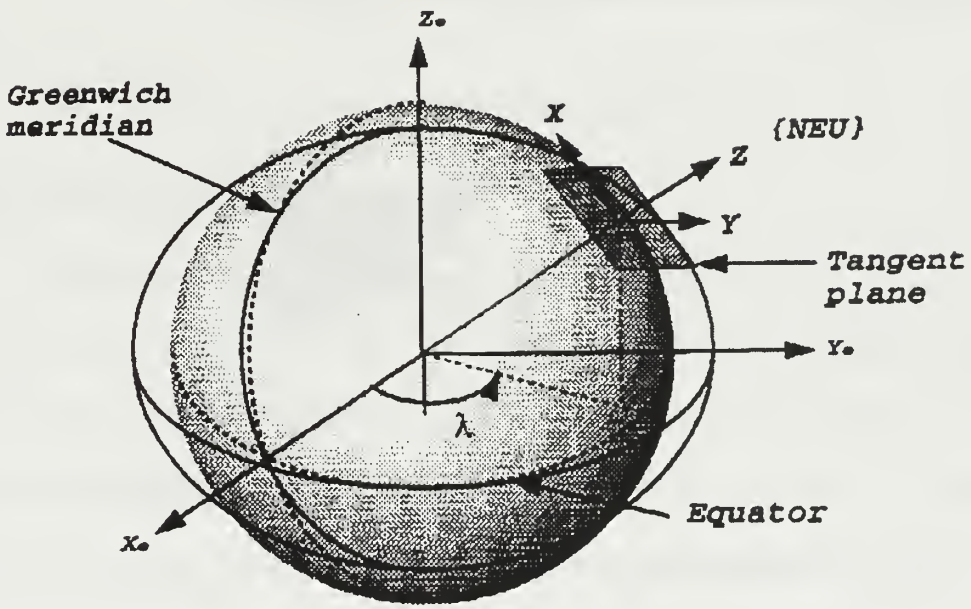


Figure 2.1: Local Tangent Plane From Ref. [3].

The point of intersection of the plane with the surface of the earth is defined to be the origin of the LTP system. The X-axis points toward true North. The Y-axis is perpendicular to the X-axis and points toward true east. The Z-axis is perpendicular to the defining plane of the system, away from the center of the earth. This particular definition represents a North-East-Up (NEU) orientation. The orientation of the frame

can be specified in a number of ways: East-North-Up (ENU) or North-East-Down (NED). [Ref. 2]

2. Body-Fixed Frame

The body-fixed frame is a right-hand orthogonal coordinate system with its origin located at the vehicles center-of-gravity (cg). The X-axis points forward through the aircraft nose. The Y-axis points towards the right wing tip and the Z-axis is perpendicular to the X-Y plane pointing downward as shown in Figure 2.2.

Figure 2.2: Body Fixed Frame From Ref. [3].

3. Wind Frame

The wind axis coordinate system, {w}, is defined as the coordinate system that results when the body-fixed x_b axis is aligned with the relative wind. This axis will not normally be aligned with the body coordinate system since the aircraft flies with an angle of attack, α, and can have a sideslip β as shown in Figure 2.3.

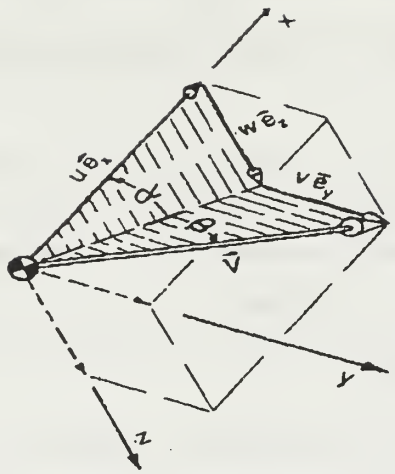


Figure 2.3: Alpha and Beta Angles from Ref. [4].

D. COORDINATE TRANSFORMATION

1. Inertial to Body

Euler angles are used to define the orientation of two coordinate systems with respect to each other. These angles define how the body-fixed coordinate system {b} is oriented with respect to the local tangent plane or universal frame {u} in terms of roll ϕ, pitch θ and yaw ψ as shown in Figure 2.4. The Euler angles enable the transformation or rotation of vector information to and from these two coordinate systems.

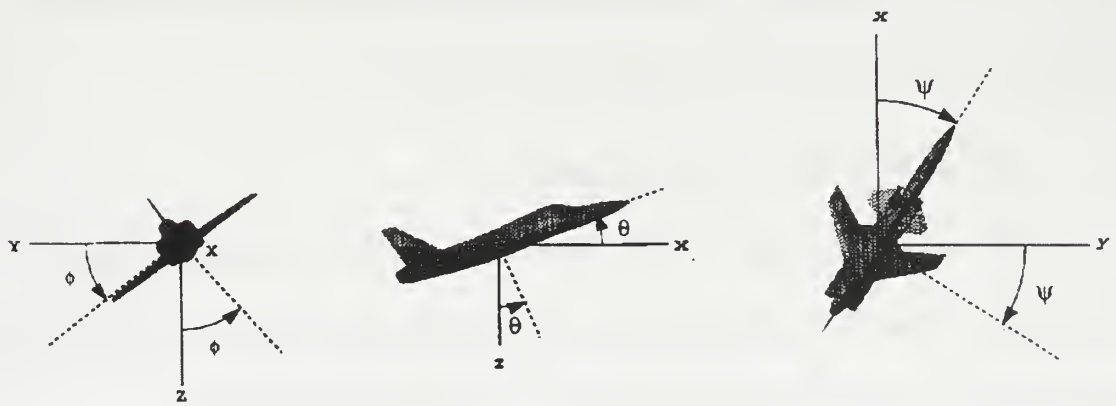


Figure 2.4: Euler Angles Phi, Theta and Psi From Ref. [3]

This rotation matrix ${}^{u}_{b}C$ is shown in Equation 1 and is used to transform vehicle velocity with respect to the air mass (indicated airspeed) from body-fixed frame to inertial frame.

$$\begin{bmatrix} x_u \\ y_u \\ z_u \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi+\sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi+\cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi+\sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi+\cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \quad 1$$

2. Wind to Body

As shown in Figure 2.3 it is possible to define the orientation of {w} to {b} with the use of the angles α and β. Using these angles it is possible to construct the transformation from {w} to {b}. This transformation is performed in the same fashion as the Euler angle transformation discussed earlier. The rotation matrix, ${}^{b}_{w}C$, is a function of α and β and is expressed as:

$${}^{b}_{w}C = \begin{bmatrix} \cos\alpha\cos\beta & -\cos\alpha\sin\beta & -\sin\alpha \\ \sin\beta & \cos\beta & 0 \\ \sin\alpha\cos\beta & -\sin\alpha\sin\beta & \cos\alpha \end{bmatrix} \quad 2$$

For the purposes of this thesis this rotation is assumed to be identity due to the fact that the FROG operates in a flight regime where the angle of attack and sideslip are negligible. This assumption will be validated in Chapter V.

E. LINEAR POSITION AND VELOCITY

Euler angles measured by IMU can be used to obtain the rotation matrix ${}^{u}_{b}C$. This enables the use of the relationship between linear position resolved in the LTP or universal frame {u} and velocity resolved body-fixed frame discussed next.

Let ${}^{u}V_{cg}$ be the velocity of the aircraft center-of-gravity with respect to the universal frame resolved in universal frame {u}. Let ${}^{w}V_{cg}$ be the velocity of the aircraft center-of-gravity (cg) with respect to the wind frame. Let ${}^{u}P_{cg}$ be the position of the aircraft cg with respect to the universal frame resolved in the universal frame and let ${}^{u}V_{w}$ be the velocity of the air mass (wind) with respect to the universal frame resolved in universal frame. Then

$$ {}^{u}\dot{P}_{cg} = {}^{u}_{w}C\ {}^{w}V_{cg} + {}^{u}V_{w} \qquad 3 $$

Equation 3 shows the relationship implemented within the filter. It is apparent that the difference of true inertial velocity and indicated airspeed is the velocity of the wind. By extension, a comparison of DGPS position, in this case true position, and position estimate derived by integrating indicated airspeed should result in an error signal that when integrated will estimate the bias between true inertial velocity and indicated airspeed, which is the velocity of the wind.

III. FILTER DESIGN REQUIREMENTS

A. DESIGN GOALS

The basic goal for the design of this filter is to improve the accuracy of the position estimate provided by the navigation system when the DGPS position update is not available. When the DGPS position update information P_{DGPS} is available, the filter should use it to update the position estimate $\hat{P}$ developed by the filter. In steady state $\hat{P}$ should equal P_{DGPS}. The improvement is provided in the form of an accurate position estimate between DGPS updates since the filter can now use the learned wind and measured vehicle airspeed to accurately estimate position.

B. BASIC COMPLEMENTARY FILTER DESIGN

Equation 3 suggests the following realization for the complementary filter shown in Figure 3.1.

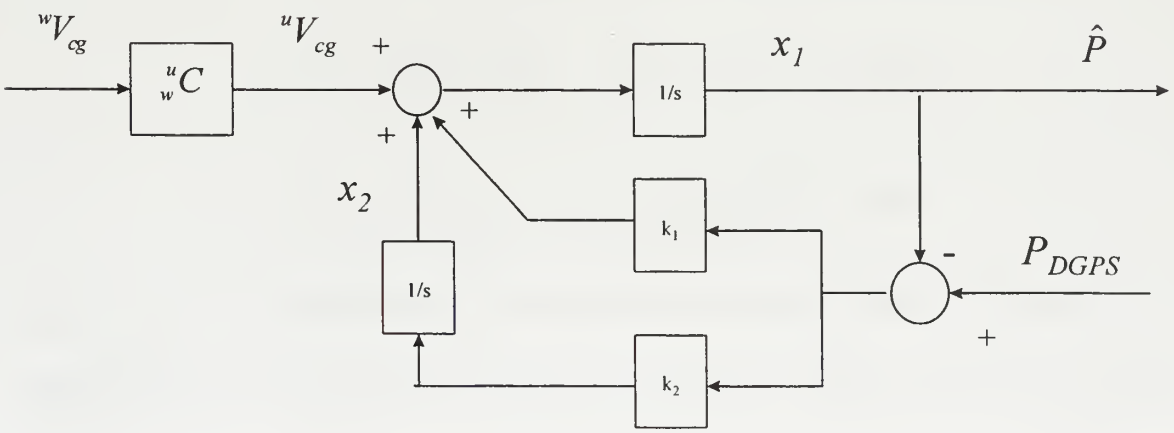


Figure 3.1: Complementary Filter Block Diagram

$$\dot{x}_1 = {}^{u}_{w}C\,{}^{w}V_{cg} + {}^{u}V_{w} + k_1\left(P_{DGPS} - \hat{P}\right)$$
$$\dot{x}_2 = k_2\left(P_{DGPS} - \hat{P}\right) \ ,$$

where: 4

$$x_1 = \hat{P}$$
$$x_2 = \hat{V}_w \quad ,$$

and k_1 and k_2 represent the filter gains. In the state space form the realization becomes:

$$\begin{bmatrix}\dot{x}_1 \\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}-k_1 & 1 \\ -k_2 & 0\end{bmatrix}\begin{bmatrix}x_1 \\ x_2\end{bmatrix} + \begin{bmatrix}k_1 \\ k_2\end{bmatrix}P_{DGPS} + \begin{bmatrix}1 \\ 0\end{bmatrix}{}^{u}V_{cg} \qquad 5$$

$$x_1 = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1 \\ x_2\end{bmatrix} \ ,$$

Using Equation 5 it is possible to then determine the transfer function of the output $\hat{P}$.

$$\hat{P}(s) = C(SI - A)^{-1}B \qquad 6$$

Completion of this matrix multiplication yields:

$$\hat{P}(s) = \frac{1}{s^2 + k_1 s + k_2}\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}s & -1 \\ k_2 & s + k_1\end{bmatrix}\left(\begin{bmatrix}k_1 \\ k_2\end{bmatrix}P_{DGPS} + \begin{bmatrix}1 \\ 0\end{bmatrix}{}^{u}V_{cg}\right) \qquad 7$$

$$\hat{P}(s) = \frac{\left(sk_1 + k_2\right)}{s^2 + k_1 s + k_2}\,P_{DGPS} + \frac{s}{s^2 + k_1 s + k_2}\,{}^{u}V_{cg}$$

From this transfer function representation and application of the final value theorem it can be shown that in steady-state the position estimate $\hat{P}$ is equal to the DGPS position P_{DGPS}:

$$\hat{P}(0) = \frac{k_2}{k_2}\,P_{DGPS} \ , \qquad 8$$

which is the desired outcome since,

$$\dot{\hat{P}} = {}^{u}_{w}C\,{}^{w}V_{cg} + {}^{u}V_{w} \Rightarrow \dot{\hat{P}} - {}^{u}_{w}C\,{}^{w}V_{cg} = {}^{u}V_{w} \,. \qquad 9$$

Therefore, it is apparent that this implementation can accurately estimate the vehicle position in steady-state due to the ability of the integrator to learn the wind (bias) and apply this correction to the measured indicated airspeed.

Selection of filter gains k_1 and k_2 was made to ensure stability and provide reasonable response time. Using these criteria k_1 and k_2 were chosen as six and nine respectively providing reasonable response time and time constant of .3.

IV. REALSIM IMPLEMENTATION

A. SOFTWARE OVERVIEW

Use of the REALSIM software installed on the UNIX workstations and its application in the design and testing of dynamic control systems has greatly simplified the workload of the control engineer. Through the use of a Graphical User Interface (GUI) shown in Figure 4.1 the designer can follow a flow diagram outlining the different software tools and components used to implement a given design. These software tools, all members of the $Matrix_x$ product family, consist of Xmath/SystemBuild, Autocode, Interactive Animation (IA) Builder, Hardware connection Editor (HCE), and utilities used to compile, link, download and run the coded control design. The use of REALSIM for the design of UAV control systems was explored in an earlier thesis project. A brief description of these applications is given next. [Ref. 5]

1. Xmath/SystemBuild

Xmath/SystemBuild includes an extensive set of design and analysis functions for the classical input/output control techniques and the modern state-space control techniques. The SystemBuild program uses a hierarchical method of organization, based on the SuperBlock concept. SuperBlocks provide a way of organizing a group of blocks that define a function into a compact form for ease of display and understanding. Through the use of this hierarchy, variable names and associated signals can be passed up and down the hierarchical structure allowing the engineer to easily track and understand what variables are and where they interacts with the model. A diagram showing the interaction of Xmath and SystemBuild is given in Figure 4.2.

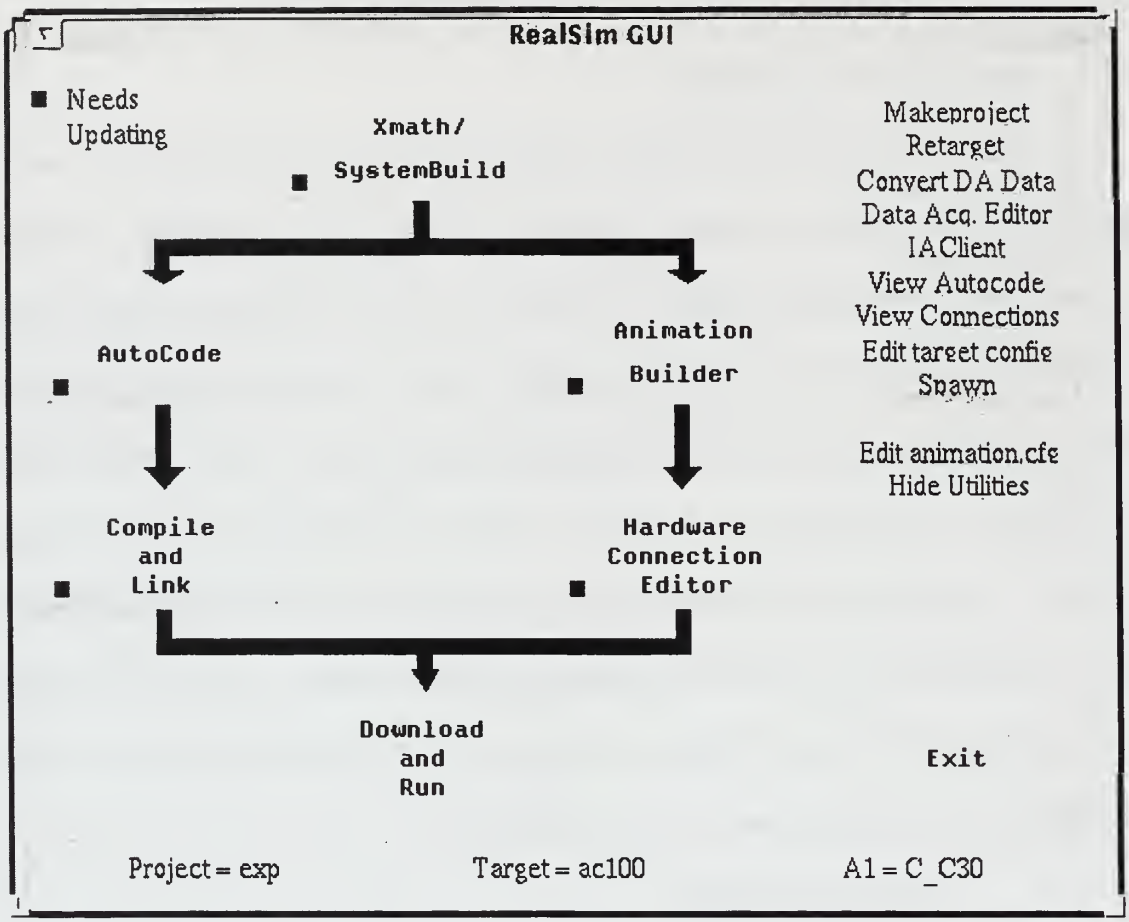


Figure 4.1: REALSIM Graphical User Interface

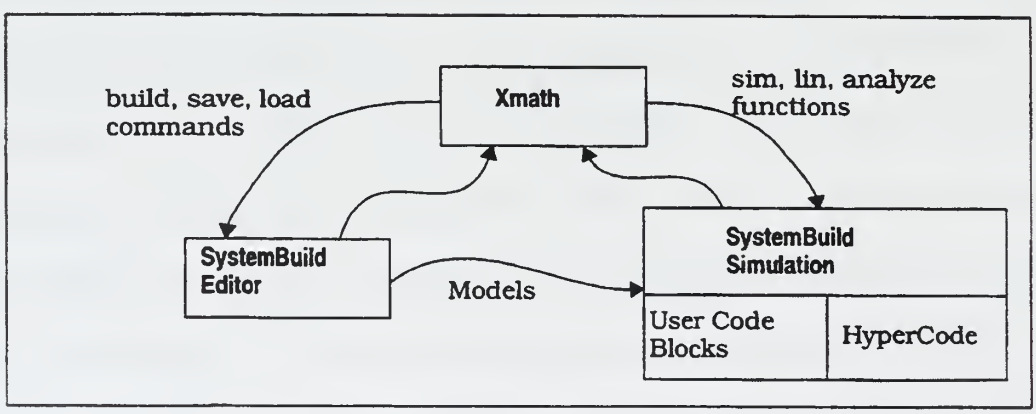


Figure 4.2: Xmath/SystemBuild Relationship

Once a given realization is drawn and labeled, there are several ways to test the model. It can be tested within Xmath/SystemBuild using the "SIM" function or by generating realtime code. The second method is preferred, since it allows for the generation of a higher-level language to conduct hardware-in-the-loop testing. To generate real-time code the user utilizes a pull-down menu on the SystemBuild GUI and selects "Generate Real Time Code". This produces a file with the model name and appended **.rtf** extension. This Real Time Code is a top level Input/Output code used by the AutoCode program to produce a higher-level code such as C. [Ref. 6]

2. AutoCode

An integral part of the quick design and testing of a controller is the ability to generate high-level code such as ADA or C automatically. AutoCode fulfills this task by generating optimized code from a library of standard functions and calls. The Realsim package in the Avionics lab utilizes the C code module. Code generation is accomplished by first ensuring the **target_config.cfg** file settings are correct and then clicking the **AutoCode** button on the main Realsim GUI shown in Figure 4.1. The **target_config.cfg** file is created using the Realsim **retarget** utility. This designates the network computer on which the application will be compiled , downloaded and executed. Once the code generation is complete a new .c appended file is created in the working directory which will later be used to compile, link, download and run the coded design. [Ref. 6]

3. Interactive Animation Editor (IA)

Interactive Animation Editor (IA) is used to build a graphical animation and necessary user interface module to display and control desired system input/output (I/O) parameters during testing. User display and interface screens are constructed through a

drag and drop process using a library of pre-drawn gauges, strip charts, dials, switches, and other input/output devices. Custom display pictures can also be created. The "RTF Names" button loads the I/O names from the model **.rtf** file to ensure correct association of given I/O signals to their display icons. An RTF file must be loaded prior to making any connections in the IA editor. The IA connection editor is similar to that used in SystemBuild. Following the completion of a picture the user selects "Save Picture" and an appended **.pic** file is added to the working directory. This file will be used later in the Hardware Connection Editor (HCE) and link process. Note: if the SystemBuild I/O is changed, the IA Editor must be run again and connections changed to reflect the changes to the model. [Ref. 5]

4. Hardware Connection Editor

The hardware connection editor is used to associate external inputs and outputs in the SystemBuild model with either external I/O devices or the I/A module. This is accomplished using two editor screens, one for external input and one for external outputs.

The HCE reads the external I/O from the **.rtf** file having the same name contained in the current target information. The HCE determines if a local copy of the **c_c30.hce** is present and, if so, reads the file. If not, a default is read out of the AC100 or AMERICA directory. This file informs the HCE what type of external I/O devices are present in the target AC100 computer. A detailed explanation of these screens and their uses can be found in the $Matrix_x$ Core Manuals, [Ref. 5].

5. Compile and Link

Once the desired inputs and outputs are connected, the design needs to be compiled and linked to the C30. The Realsim software will attempt to connect via ftp with the targeted computer. Once the connection is made, the required source C files will be transferred to the target computer for remote compiling and linking. The compiler generates object code from the **.c** source file, and the link creates a C30_DSP executable code from the object code. If there are other C files required for the project for compiling and linking there should also be a file in the working directory named **sa_user.cmd**. In this file there should be a line that reads **COMPILE <<filemane>>.c** and one reading **LINKWITH<<filename>>** for every all external user produced C-code files. [Ref. 6]

6. Download and Run

When this program is selected on the GUI, Realsim will attempt to connect to the target computer. Once the connection in made, the executable code is loaded into the C30 memory where it is prepared to run. The IA module is then brought on the screen along with an IA Client. The IA Client is shown in Figure 4.3.

The first IA Client button, "Start Controller", will start the model if the model has jest been loaded and not yet run. It will stop the model if it is currently running and restart it if it had previously been stopped. The second "Hardware Reset" button causes the Realsim controller to immediately reset and exits the IA Client. This action clears the C30 memory and returns the target computer to a ready status. The "Exit Graphics" exits the IA Client without rebooting the target controller. This is a software reboot only, which stops the model and terminates the ftp connection. This button is not recommended for use due to its inability to stop the model from running on C30. If download and run is selected again by the original client which started the model, it will

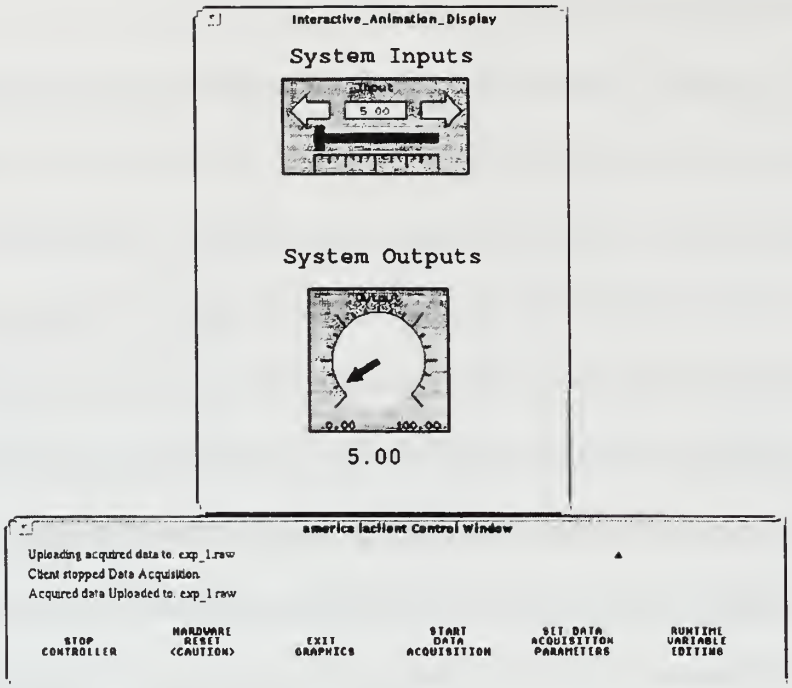


Figure 4.3: IA Client Control Window

ask if you wish to reconnect the model. If a different client attempts to log on to run a model it will ask if the current model should first be terminated. Therefore it is always preferable to terminate the model by selecting stop controller and hardware reset. The "Start Data Acquisition" button starts and stops data acquisition. Each time Start Data Acquisition is selected data signals selected via the Data Acquisition Editor are recorded and stored in a **project_name.raw** file. Following flight test, these files are converted to **project_name.dat** files for use in system analysis. Data acquisition should be stopped before stopping the controller. Should the controller be stopped before the data acquisition stops the acquired data will not be saved to a file. The "Scale Frequency" button allows the user to set certain data acquisition parameters. Other data acquisition features can be invoked from the menu located in the upper right corner of the Realsim GUI and selecting "Show Utilities". These utilities enable the user to capture specific

data within the model for use in Xmath. More information can be found in the Matrix$_X$ Core Manuals.

B. FILTER INPUTS

1. Velocity

Measured FROG airmass velocity is available from the existing on-board instrumentation and within the SystemBuild hierarchy developed earlier. Calibration of this data is addressed in Appendix A. The velocity, in units of feet-per-second, is provided to the **GPS_filt** Superblock in Figure 4.4 through an external input in the form:

$$ {}^{w}V_{cg} = \begin{bmatrix} V_T \\ 0 \\ 0 \end{bmatrix} \qquad 10 $$

where V_T is the aircraft indicated airspeed. This velocity is then resolved in the inertial frame in the "**b_to_i**" block (14) through the use of the rotation matrix ${}^{u}_{b}C$ utilizing the Euler angle inputs.

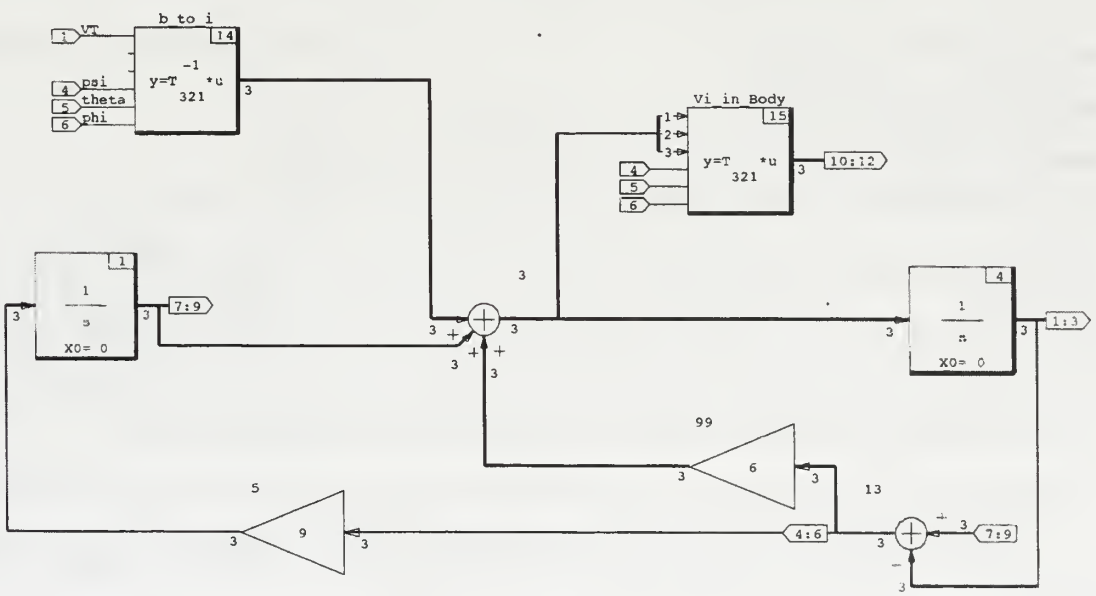


Figure 4.4: Continuous Position Filter **gps_filt**

2. Euler Angles

Estimates of the Euler angle psi are generated in the **Sensor Filters** SuperBlock shown in Figure 4.5 using DGPS heading (hea), IMU angular rate information [*p, q, r*] and GPS inertial velocity. Heading estimates are computed through complementary filtering of **psi_dot,** generated in the **L_dot_eq** SuperBlock, and DGPS heading **hea**.

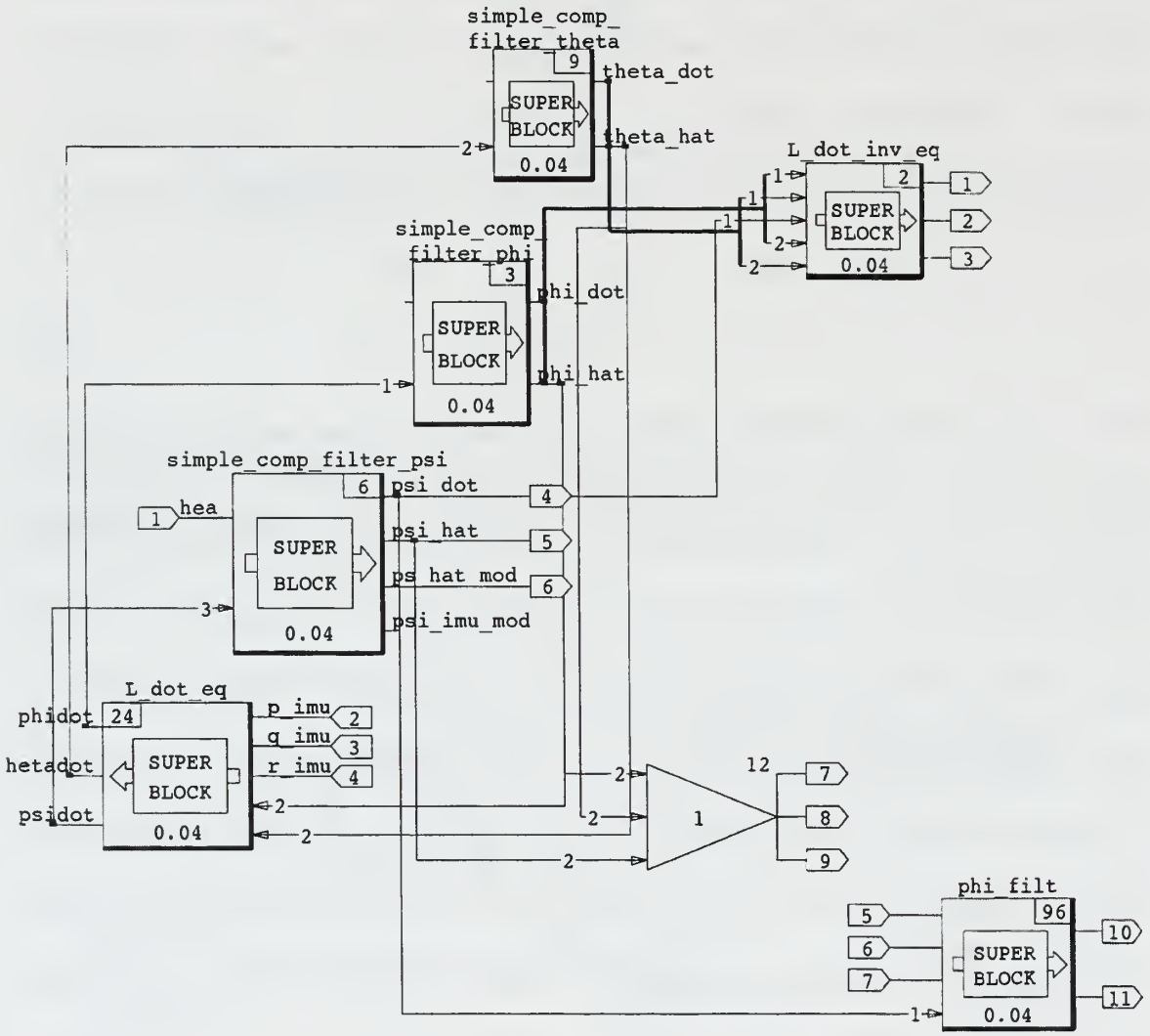


Figure 4.5: Euler Angle Filter Bank in Sensor Filters SuperBlock

This project will address the determination and evaluation of psi in the implementation of the filter and the generation of a low frequency phi estimate.

3. DGPS Position

The task of incorporating DGPS into the FROG navigation system was accomplished in an earlier thesis [Ref. 1]. DGPS position resolved in Local Tangent Plane is taken as input in units of feet north, east and down.

C. FILTER OPERATION

The goal of this project was to implement a filter, using available INS and DGPS information, that would provide accurate position estimates in the inertial frame when DGPS was not available, i.e. between the one second updates. It is essential that all computations be made with data resolved in the same reference frame. This is accomplished in the filter through the use of the available Euler angle estimates and application of the transformation matrix to resolve velocity in the inertial frame {u}. Another essential operation is to compare the estimated position with the DGPS position when available. The important point here is to realize that a new error signal is available only at the same rate as the DGPS update. This error signal is determined by subtracting estimated position from DGPS position:

$$e = \left(P_{DGPS} - \hat{P}\right). \tag{11}$$

In the absence of a DGPS update the feedback gains k_1 and k_2 must be disconnected. This results in the position estimate being computed based on an inertial velocity that, until the integrator has "learned" the bias (wind), will be erroneous. The discrete version of the feedback portion of the position filter shown in Figure 4.6 implements the logic needed to turn the feedback gains on and off. This is accomplished by comparing the current P_{DGPS} input to the previous value, which is completed in summing block (93). When that result is non-zero, i.e. there has been a DGPS update,

the gains are connected and the latest position is used to improve the estimates. This switching logic is implemented using a combination of coded (C-code) blockscript and data path switches.

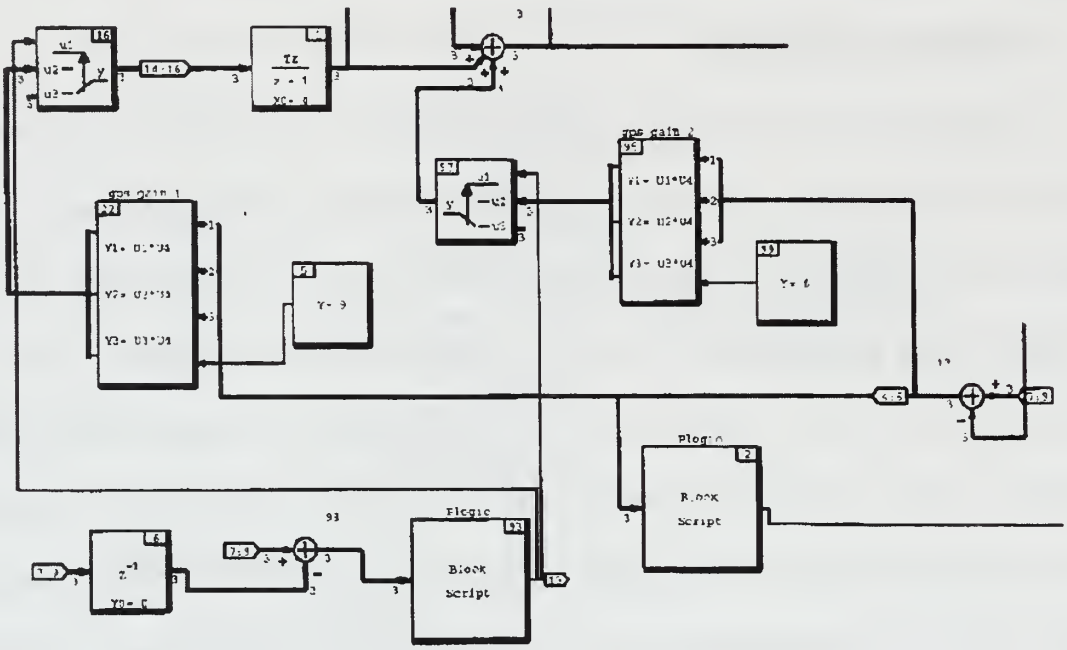


Figure 4.6: Discrete Position Update and Gain Switching Implementation

The blockscript, **Plogic**, takes the result of the summing block (93) difference of the current and previous P_{DGPS} value and computes its norm. The code generates a boolean output which is high when the norm is non-zero (DGPS has updated) and low otherwise. This logic triggers the data path switch to either connect or disconnect the feedback gains.

Learning the bias between the indicated airspeed and the estimated inertial velocity, which is the wind estimate, enables the filter to develop an accurate estimate of

the inertial velocity and position. It is apparent that in steady state the filter provides accurate estimates of the position and its derivative, inertial velocity.

D. FILTER OUTPUTS

1. Position Estimate

The position information, DGPS or position estimate, is taken as the three channel output of the data path switch and labeled **North_ltp_filt**, **East_ltp_filt** and **Down_ltp_filt** corresponding to outputs ten, eleven and twelve of Block (96) in **gps_filt**. This position information is then taken to the highest upper-level SuperBlock, **Process 1**, where it is then sent through the **output variables** gain block and made available as an external output for use in data acquisition, flight test and animation display.

2. Wind Estimate

The wind estimate is available at the output of the data path switch block (16) in Figure 4.7 where it is labeled **wind_bias_n**, **wind_bias_e** and **wind_bias_d** and taken as output seven, eight and nine. From the **GPS_Filt** SuperBlock it follows the same path as the position information as an external output. This information is essential in evaluating the performance of the filter during flight test and is discussed in Chapters V and VI.

3. Inertial Velocity

Inertial velocity is available at the output of the summing junction block (3) of the GPS_Filt SuperBlock in Figure 4.7 where it is labeled **V_i_n**, **V_i_e** and **V_i_d** and taken as output 28, 29 and 30 respectively. This velocity information follows the same path as the position and wind information as an external output.

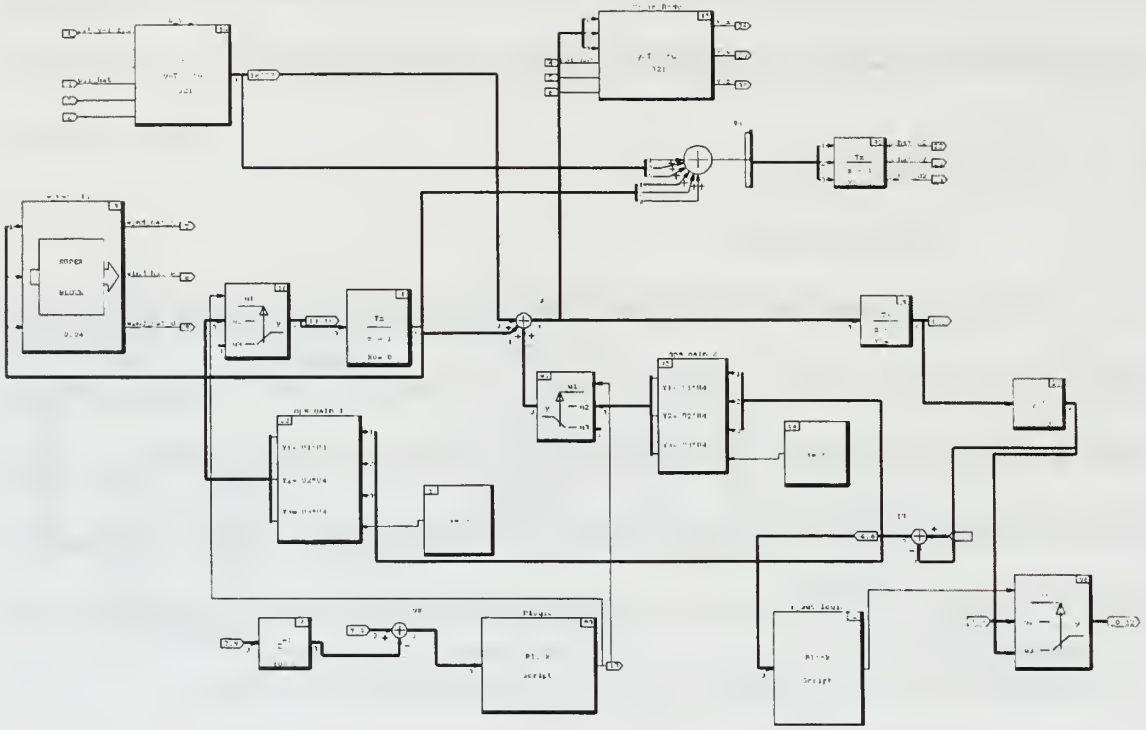


Figure 4.7: Discrete Position Filter

V. SIMULATION

A. FROG MODEL

The development of a dynamic model of the FROG UAV was accomplished in a precious thesis project [Ref. 7]. Two of the three inputs required for the implementation of this filter are available as outputs from this dynamic model. The parameters of body referenced velocity resolved in wind frame and the Euler angles ψ, θ and ϕ are computed and available for use as inputs to the position filter. The FROG model also needed control deflection information which was available from a previously developed model of the autopilot used on the FROG [Ref. 8]. Future references to the FROG model will include both the FROG and Autopilot models.

The initial conditions for the FROG model are established in the state space **Initial Conditions** block (97) within the FROG SuperBlock shown in Fig. 5.1. The input vector, x_0 to x_{11}, establishes the initial values of the parameters used in the FROG model which are $[u_0, v_0, w_0, p_0, q_0, r_o, \phi_0, \theta_0, \psi_0, Px_0, Py_0, Pz_0]$

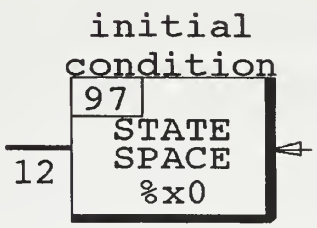


Figure 5.1: Frog Model Initial Conditions Block

These parameters are defined below:

$$\begin{bmatrix} u_0 \\ v_0 \\ w_0 \end{bmatrix} \equiv \quad \text{Inertial Velocity with respect to \{u\} resolved in \{b\}}$$

$$\begin{bmatrix} p_0 \\ q_0 \\ r_0 \end{bmatrix} \equiv \quad \text{Angular Rates of Roll, Pitch and Yaw resolved in \{b\}} \qquad 12$$

$$\begin{bmatrix} \phi_0 \\ \theta_0 \\ \phi_0 \end{bmatrix} \equiv \quad \text{Initial Euler Angles Defining Orientation of \{b\}}$$

$$\begin{bmatrix} Px_0 \\ Py_0 \\ Pz_0 \end{bmatrix} \equiv \quad \text{Initial Position in LTP in N-E-D Orientation}$$

Velocity is input in units of feet-per-second, angular rates in radians-per-second, angles in radians and position in feet relative to LTP origin. The velocity components of the wind are inputs to blocks (94) and (95) of the **dynamics_euler** SuperBlock shown in Figure 5.2. Block (94) contains the simulated wind velocity in {u} frame which is output to **wind_u2b** block (95). Here it is resolved in {b} frame for use in the **aero forces and moments** block (96). Use of these initial conditions and wind parameters will be discussed in Section C of this chapter.

B. DGPS UPDATE

Modeling of the DGPS update rate was accomplished in the **gps_updt** block(98) within the **pos_updt_mdl** SuperBlock shown in Figure 5.3. The **gps_updt** SuperBlock receives true position, simulating DGPS position information, from the FROG model.

For continuous time simulation the position information passes through block (4) and is unaltered. For discrete time various update rates were simulated by changing the sample interval of the discretized **gps_updt** SuperBlock. This is easily accomplished in SystemBuild and will be discussed in section D.

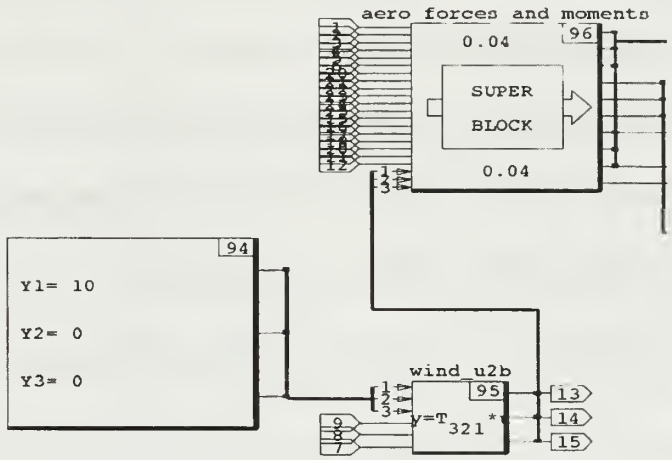


Figure 5.2: Wind Input to Dynamics_Euler SuperBlock

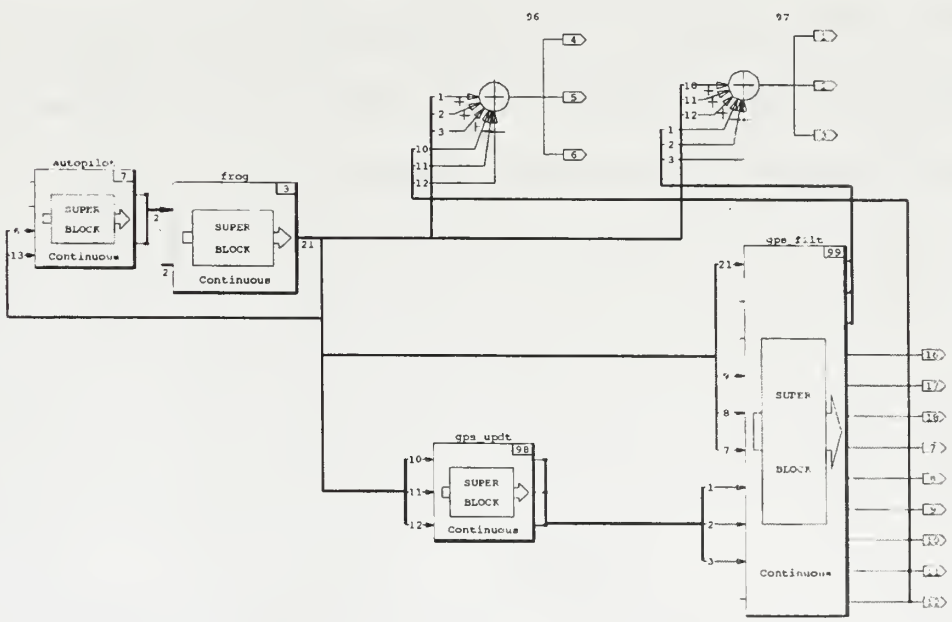


Figure 5.3: Continuous **pos_updt_mdl** SuperBlock

C. CONTINUOUS TIME SIMULATION

Continuous time simulation of the mechanization developed in Chapter III was completed using Xmath/SystemBuild with the continuous version of **pos_updt_mdl** SuperBlock shown in Figure 5.3. This SuperBlock contained the **frog**, **autopilot**, **gps_updt** and **gps_filt** SuperBlocks. Initial conditions for each simulation were input in the appropriate SuperBlock as described in Section A of this chapter. The basic procedure for the continuous time simulations was to run the **pos_updt_mdl** with varying sets of initial condition and wind parameters for each of ten **gps_filt** filter gains from one to ten. Outputs of each simulation included but not limited to:

- Position Error $\left(P_{DGPS} - \hat{P}\right)$ in units of feet North, East and Down LTP
- Inertial velocity error $\left({}^{u}V_{cg} - {}^{u}\hat{V}_{cg}\right)$ in units of fps North, East and Down LTP
- Wind (bias) velocity in units of fps North, East and Down LTP

A sample of the position error and wind estimate results from a continuous time simulation are shown in Figure 5.4. All simulations used initial conditions of [u=80, r=.1, phi=.2] with all others being zero. Filter gains k_1 and k_2 were nine and six respectively for all simulations as discussed in Chapter III.

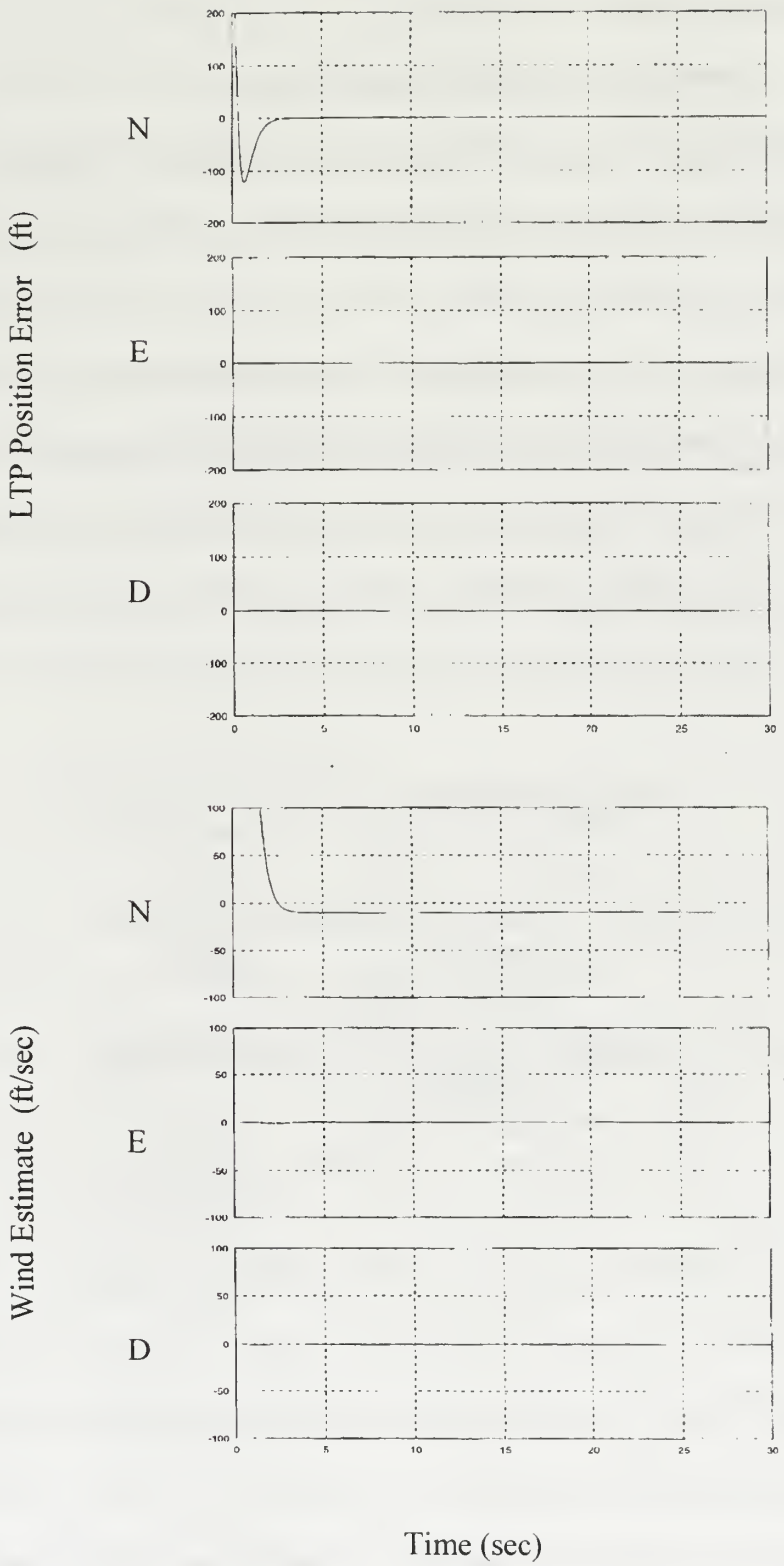


Figure 5.4: Continuous Simulation Position Error and Wind Estimate

D. SYSTEM DISCRETIZATION

1. Continuous to Discrete Transformation

Transformation of the continuous time **pos_updt_mdl** SuperBlock to discrete time was accomplished using the **Transform SuperBlock** utility in SystemBuild. This enables the user to automatically transform SuperBlocks from continuous to discrete time or from one discrete rate to another. Invoked from the **Build** menu in SystemBuild, the **Transform SuperBlock** dialog allows multiple SuperBlocks to be transformed together and provides detailed control over how the block parameters are affected during transformation. During this transformation all interconnections and labeling information is preserved. A continuous to discrete transformation is shown in Figure 5.5.

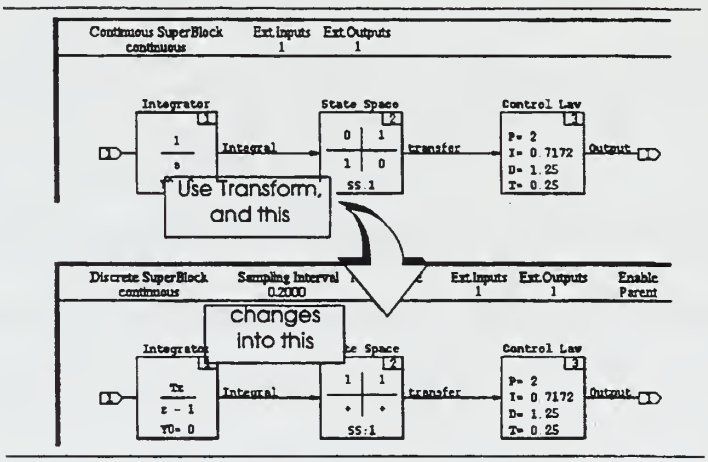


Figure 5.5: Continuous to Discrete Transformation

The implications of transforming from continuous to discrete vary depending on the type of blocks in the model: [Ref. 9]

- Algebraic, Logical and other blocks without dynamics or memory are unaffected

- Most dynamic blocks (time delay and integrators) maintain their coefficients, except for the compensation for the new sampling interval (T).
- State Space, NUM/DEN and Gain/Zero/Poles blocks require new coefficients and are transformed from continuous to discrete using Tustin's (trapesoidal) rule.

Transformation from continuous to discrete using **Transform SuperBlock** launches a dialog showing a SuperBlock Catalog listing all loaded SuperBlocks and their sample rates. From this list the user may select a hierarchy, all or individual SuperBlocks to transform. The dialog prompts for four pieces of information that control the transformation:

- New sample interval (0 = cont)
- New first sample (skew)
- Discrete to Discrete transform (rate only or rate and block coefficients)
- Transform initial conditions (of state space blocks)

Clicking the **Transform** button completes the process, updating the new rates and overwriting the old SuperBlocks.

2. Discrete Simulation

Using the method described above, the **pos_updt_mdl** SuperBlock was transformed from continuous to discrete with a sampling interval of .04 seconds (25 Hz). The initial discrete simulation was run with all SuperBlocks sampled at the same rate of 25 Hz to establish a baseline for comparison of the multi-rate characteristics of the filter. On successive simulations the **gps_updt** SuperBlock was transformed to slower sample rates to simulate DGPS update performance.

Transforming the **gps_updt** Superblock to a slower sampling rate than the rest of the system results in the multi-rate system. This was done to simulate the DGPS position usually being available once a second with infrequent delays of 2-3 seconds. It is this characteristic that requires the filter feedback gains to be turned off when the DGPS position information is not available. Consequently, the DGPS update logic blocks (6, 93 and 98) and the data path switches (blocks 16 and 97), used to connect and disconnect the **gps_filt** feedback gains were incorporated to the system. The final discretized filter is shown in Figure 5.6.

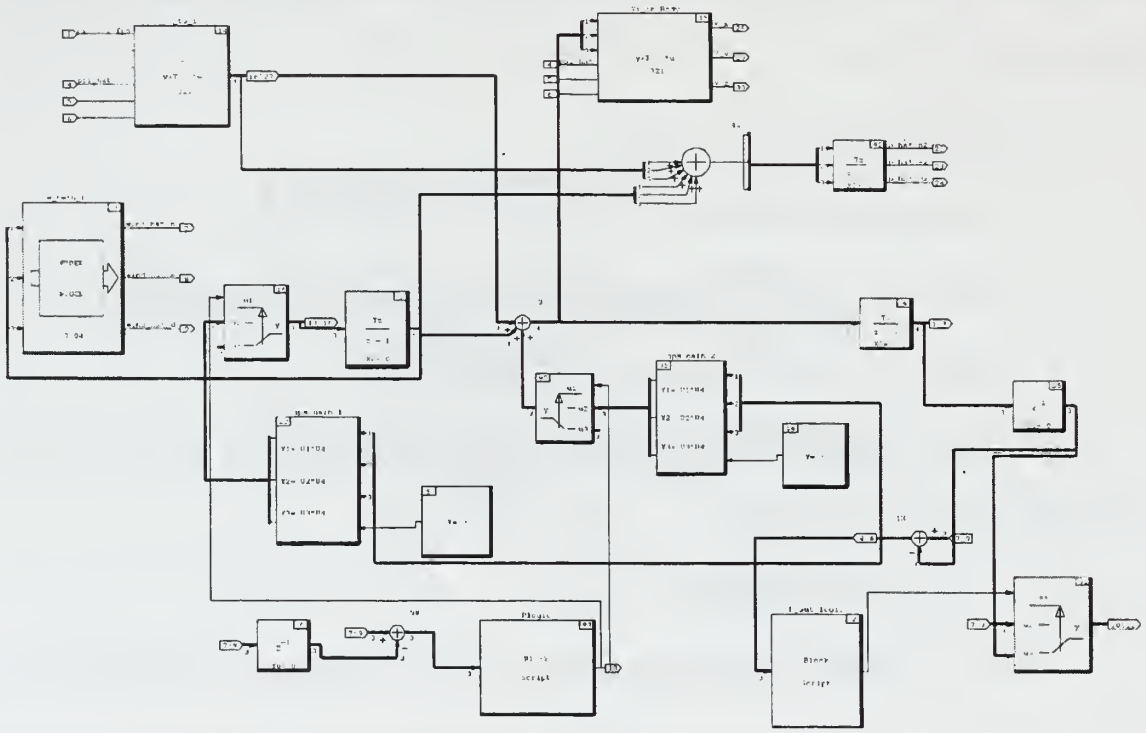


Figure 5.6: Discrete **gps_filt** SuperBlock

Initial conditions for these discrete simulations were {u=80,r=.1,phi=.2,P_0=880}. Those parameters not listed were zero. Position error (P_E) and wind estimates outputs in Figure 5.7 shows the results of the discrete simulation of the system with DGPS update rate of 1 Hz.

Figure 5.7: Position Error and Wind Estimate (gps_updt=1 sec)

3. Effects of Angle-of-Attack and Sideslip

Having established the implementation of the discrete time filter in simulation it was now necessary to examine the errors introduced by the assumption that the body and wind frames are essentially the same for the flight envelope of the FROG. This was examined using the values for angle of attack and sideslip developed in the FROG model used in the discrete simulations. These α and β quantities were used as inputs to the rotation matrix ${}^{b}_{w}C$ defined in Chapter II to transform the measured indicated velocity from wind to body frame. This velocity was then used as input to the position filter. The filter position estimates were compared to the results from the previous simulations where alpha and beta were assumed to be zero. Simulations were run using wind speed inputs to the FROG model of 0 to 50 feet-per-second made up of north and east components.

Angle-of-attack measurements from the FROG model fell in a range of -.02 to +.042 radians and the sideslip values ranged from +/- .1 radians. As expected the difference between position estimates using the actual {w} to {b} rotation matrix and those generated using the assumption of that matrix being identity were less than one percent.

VI. FLIGHT TEST

A. FLIGHT TEST SETUP

Proper functioning of the position filter relies on the availability of accurate LTP position data from the DGPS system. The configuration of two software items must be checked prior to take-off to ensure proper DGPS function. For the software, the coordinates of the LTP origin must be set and confirmed in both RealSim and the DGPS receiver coding. Coordinates within the RealSim software were verified using the following procedure. From the GUI **Master** page shown in Figure 6.1, select the **LTP Navigation** page. From this page, shown in Figure 6.2, view the coordinates displayed for latitude, longitude and geoide height of the LTP origin.

Figure 6.1: IA Client Master Page

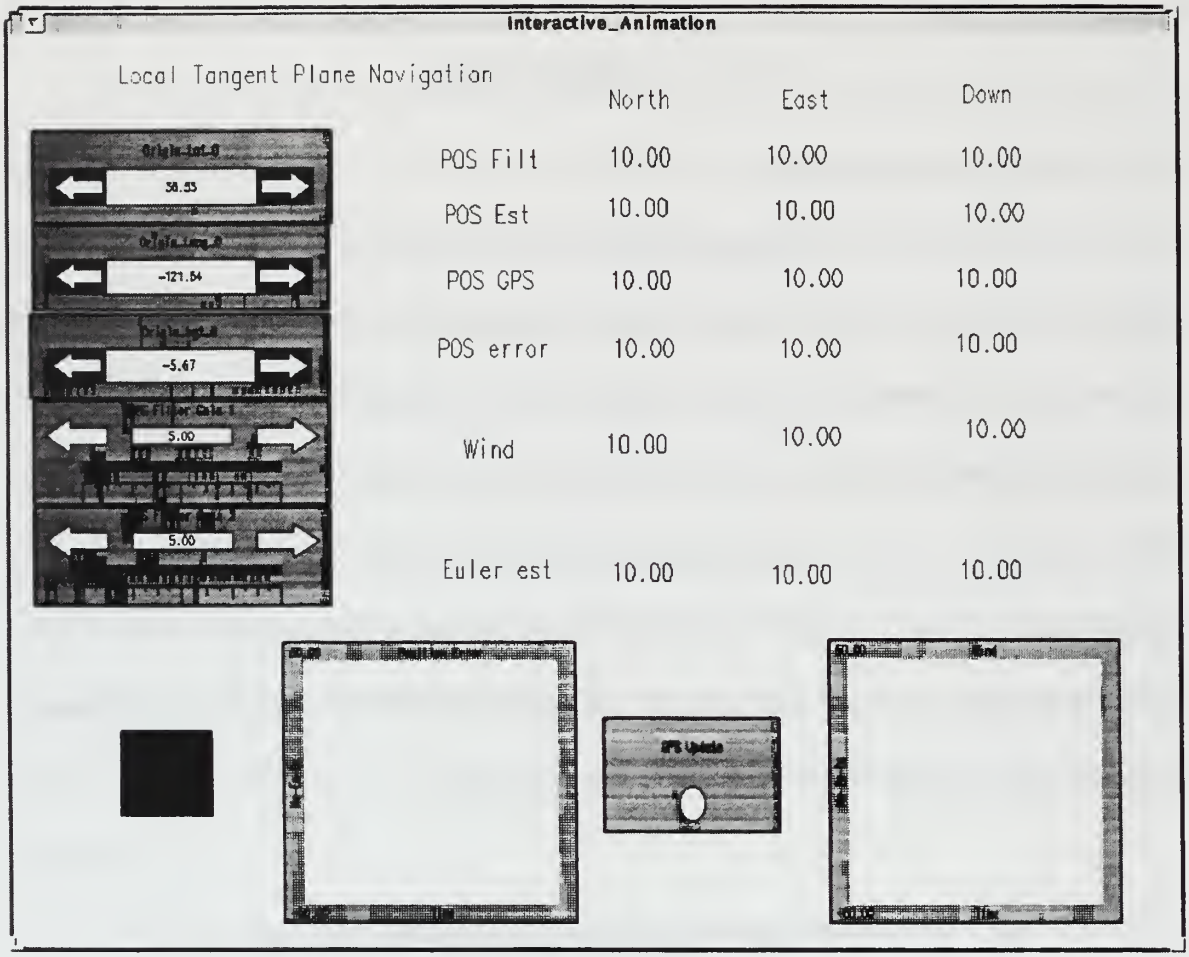


Figure 6.2: IA Client LTP Navigation Page

The coordinates for the surveyed spot at Chular (RC) airfield, where the differential GPS antenna is located during flight test, are set as default values in the **Orig_lat_0**, **Orig_long_0** and **Orig_hgt_0** icons on the **LTP Navigation** page. Modifying the coordinates is accomplished by double-clicking the upper right hand corner of the respective display box to open the data input menu. From this menu the initial values, range of accepted values and other variable parameters may be changed. The sequence is completed by selecting the "Done" button. Initial coordinates within the DGPS receiver are checked and modified using the luggable PC or auxiliary laptop

notebook computer with the "GPS40" program. Specifics on this procedure are outlined in [Ref. 1].

B. FLIGHT TEST PROCEDURE

1. Data Acquisition

Data acquisition during flight test was accomplished using the Data Acquisition utility described in Chapter III. This utility enables the user to record essential SystemBuild input and output signals for use in system analysis. Figure 6.3 lists data recorded during flight test with the Data Acquisition Utility:

	Input Signals	Output Signals
1	ax_imu	psi_hat
2	ay_imu	theta_hat
3	az_imu	phi_hat
4	p_imu	north_ltp_hat
5	q_imu	east_ltp_hat
6	r_imu	down_ltp_hat
7	phi_imu	north_ltp_err
8	theta_imu	east_ltp_err
9	psi_imu	down_ltp_err
10	north_ltp	wind_bias_n
11	east_ltp	wind_bias_e
12	down_ltp	wind_bias_d
13	vel	gps_log
14	hea	
15	pit_vel_fps	
16	ail_pwm	
17	elev_pwm	

Figure 6.3: Signals Recorded Using Data Acquisition Utility

2. Preflight Checks

Following the calibration of the D_to_A converter the **LTP Navigation** page was selected from the GUI master page. **Start Controller** was selected and the **Pos GPS** readings and **GPS Update** icon were checked to ensure DGPS updates were being received. **Pos Est** and **Pos err** readings were then checked against the **Pos GPS** values to

ensure proper function. The **Psi** display was checked by rotating the UAV 360 degrees and noting the associated heading angle in degrees. Wind values were checked to be at or near zero with the UAV stationary. Both graphical displays were checked to ensure a reflection of the position error and wind readings, respectively. All LTP position, wind and angle estimate readings were monitored as the UAV was taxied from the preflight area to the runway in preparation for takeoff.

3. Flight Test Conduct

Four test runs were planned and executed at the Chular (RC) Airfield. On each two to three minute run the filter gains were increased so that a range of filter performance data would be accumulated with gains values of two, five, eight and ten respectively. Flight paths test runs consisted of steady turning circles approximately 4000 feet in diameter and constant throttle settings. Each of the last two runs were successive level runs north and south along runway heading with wings level lengths of approximately 2000 feet and associated reversals at constant throttle settings.

The Data Acquisition utility was started approximately 3 seconds prior to the beginning of each run and stopped immediately upon run completion to record the signals discussed in part One of this section. The UAV pilot maintained control throughout the flight test.

C. DATA ANALYSIS

The data recorded using the Data Acquisition utility was converted from the **.raw** to **.dat** form as discussed in Chapter IV. These multiple .dat files made up of approximately 180 seconds of flight test data signals were concatenated and saved to a **.xmd** file for use in Xmath/SystemBuild. It was now possible to analyze the flight test

data by driving the filters and models with the necessary real-time signals acquired during flight. This analysis was conducted using the Xmath and SystemBuild tools. Procedures were similar to those used in the simulation of the discrete time model.

1. Heading Data Analysis

Prior to analyzing the performance of the position filter it was first necessary to determine which heading signals would be most suitable for use in the navigation and position filters. Heading information was available from both the IMU and GPS as the signals **psi_imu** and **hea** respectively. Psi_imu and hea were inputs nine and 14 to the **Sensor Filters** SuperBlock where either could be used in both the **simple_complementary_filter_psi** SuperBlock to generate an estimate of psi and as an input to the **gps_filt** SuperBlock for use in the rotation matrix from {b} to {u}. Samples of these two signals are shown in Figure 6.4.

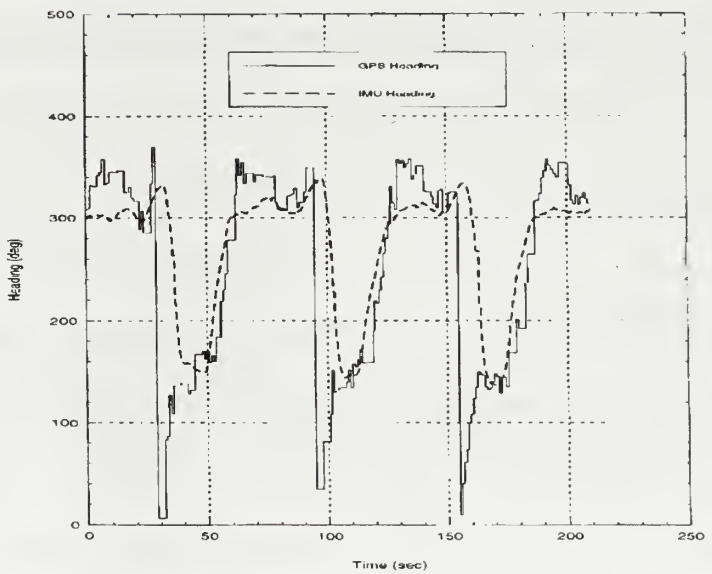


Figure 6.4: Headings **hea** and **psi_imu**

It is immediately evident that the IMU heading information, **psi_imu,** lags significantly. Analysis of the IMU heading information prior to flight test had not revealed this lag. But as shown and with all observed flight test data the IMU heading consistently lags that from the DGPS. It is clear from this comparison that despite the slow update and data latency, DGPS heading should be used as the input to both the **simple_complimentary_filter_psi** and **gps_filt** SuperBlocks for the generation of **psi_hat** and {b} to {u} rotation matrix, respectively.

2. Heading Validation

Using GPS heading, generation of heading estimate **psi_hat** was now validated using the autopilot and FROG models used in simulation. To complete this process the **frog_a_p** SuperBlock was constructed. This SuperBlock shown in Figure 6.5 was made up of the autopilot, frog and combined with the **Sensor Filters** SuperBlock to make up the **frog_val_mdl** SuperBlock. The goal of this procedure was to compare the heading estimates generated by the complementary filter with those generated by the FROG model.

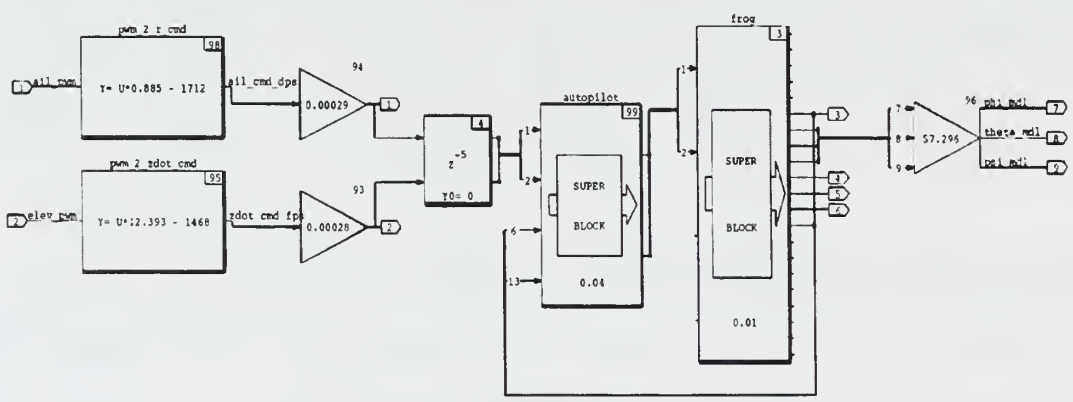


Figure 6.5: Frog Model Validation SuperBlock **frog_a_p_mdl**

The autopilot/FROG model combination was driven with yaw and climb rate command signals (**r** and **zdot**) available from the data.xmd file. The actual flight test PWM command and transmit signals used were **ail_cmd_us**, **elev_cmd_us**, **ail_pwm**, and **elev_pwm**. The distinction between the two pairs of signals is that ail_pwm and elev_pwm are the PWM commands sent from the computer to the Futaba transmitter and ail_cmd_us and elev_cmd_us are the actual PWM transmitted to the UAV autopilot. Use of the commanded signals involved incorporating a 200 ms delay in both the aileron and elevator channel prior to the autopilot input to simulate command-transmit-receive lag.

These r and zdot commands in PWM form were first converted to units of degrees-per-second and feet-per-minute respectively using the slope and x-intercept calibration information established in [Ref. 8] and finally to radians-per-second and feet-per-second for input to the autopilot model.

The heading output **psi_mdl** was compared to the filter estimated heading **psi_hat** generated using DGPS headings from several different flight test data sets. A sample of these results in Figure 6.6 show the model headings correlate well with the filter estimated headings. This further justifies the use of model generated heading in the continuous and discrete filter simulations.

3. Filter Position

Evaluation of the position filter was accomplished using the 15 inputs required by the Sensor Filter SuperBlock, previously discussed and listed in Figure 6.3. These signals were available in the data.xmd files constructed from several flight tests. Psi_hat generated from DGPS heading input to the complementary filter was used as the psi input to the position filter. Neither phi nor theta estimates from the complementary filters were validated at this time and were assumed to be zero for the purposes of filter evaluation.

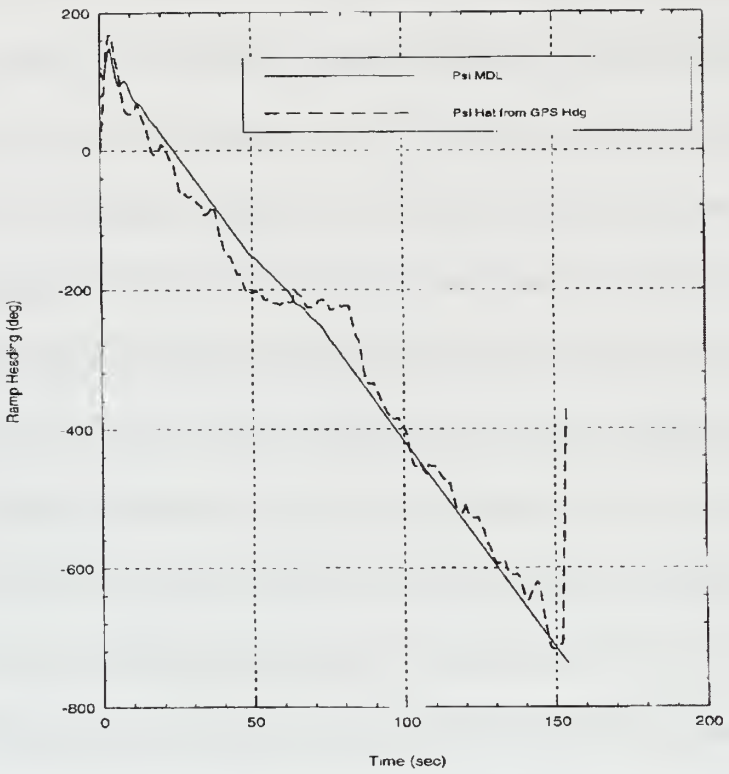


Figure 6.6: Psi_mdl and Psi_hat

In this process DGPS heading and position information were input to the filter and the resulting position and wind estimates evaluated. As with the heading validation multiple data sets from different flight tests were used providing a wide range of wind conditions with which to evaluate the system position estimate.

Filter estimated LTP position was compared to DGPS data in the north, south and down channels. Figure 6.7 shows a sample of the north channel comparison for a particular data set.

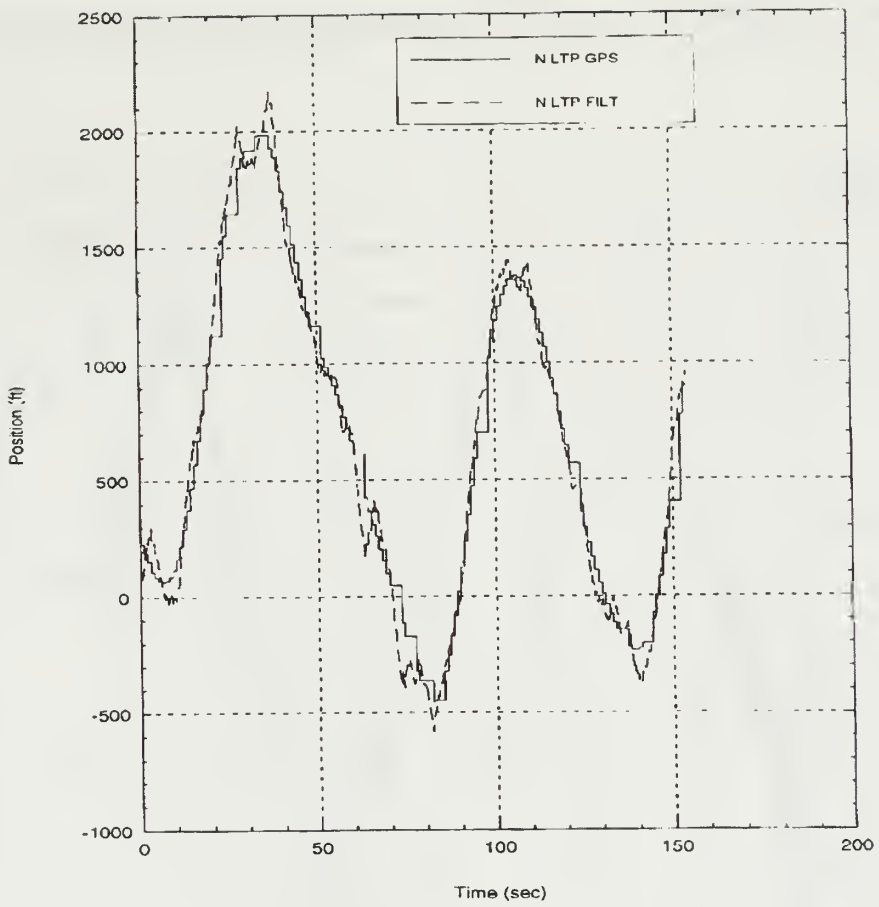


Figure 6.7: North DGPS and Estimated Position

The filter estimated position tracks the actual DGPS position well with some notable exceptions. As expected, the filter position appears as a ramp between the discrete DGPS data tracking the corner of each successive update. The estimated position spikes are caused by the absence of DGPS update information to the filter. This latency will be addressed in Chapter VII. Without the necessary position updates the filter continues to integrate the inertial velocity with no updated error signal. Lack of heading information further aggravates the estimate. As time goes on the affects of DGPS position latency are diminished due to the filter having refined its wind estimate.

Comparing these flight test results with north channel simulation shown in Figure 6.8 illustrates the effects of poor DGPS position and heading update rate on filter performance.

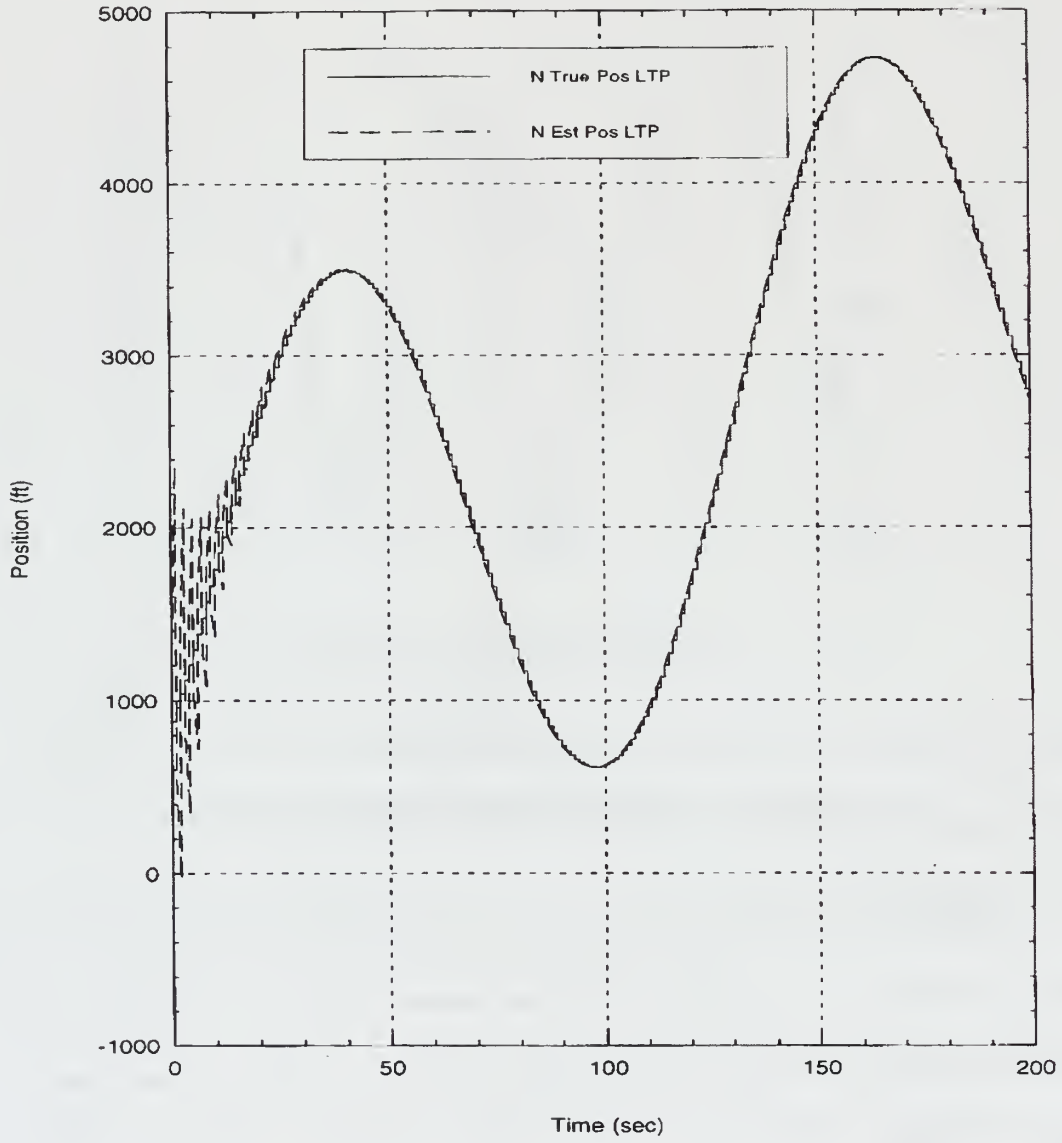


Figure 6.8: North True and Estimated Position from Simulation

Simulation results using DGPS update rate of 1 Hz without interruption clearly show the filter providing accurate position estimates within 30 seconds.

The east channel sample results shown in Figure 6.9 are much the same as those observed in the north.

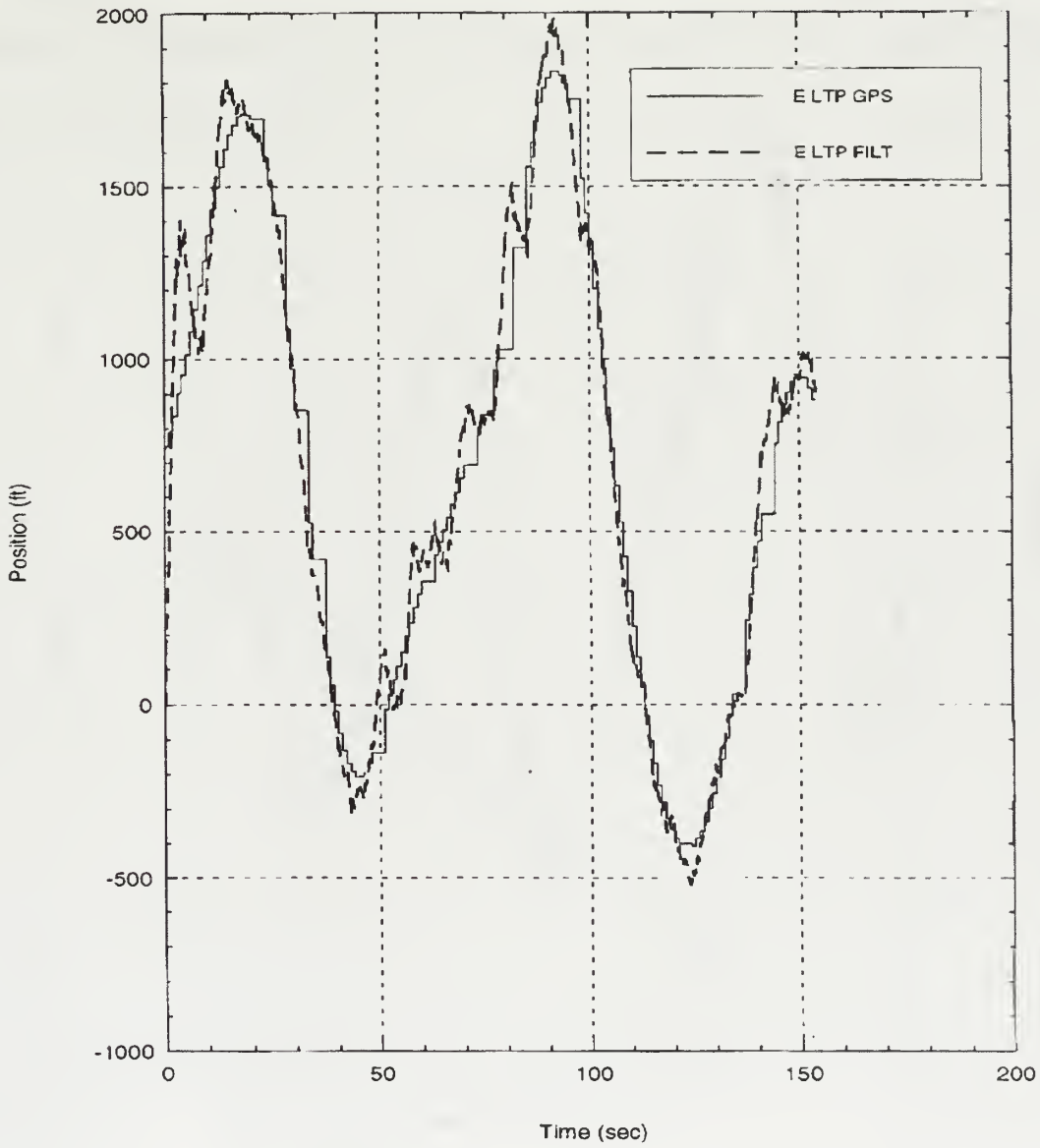


Figure 6.9: East DGPS and Estimated Position

In an attempt to demonstrate the effect of poor heading information on estimated position the heading information available from the FROG model driven by flight test

yaw and climb rate PWM commands was used as the psi input to the position filter. The north channel estimated and DGPS positions are shown in Figure 6.10.

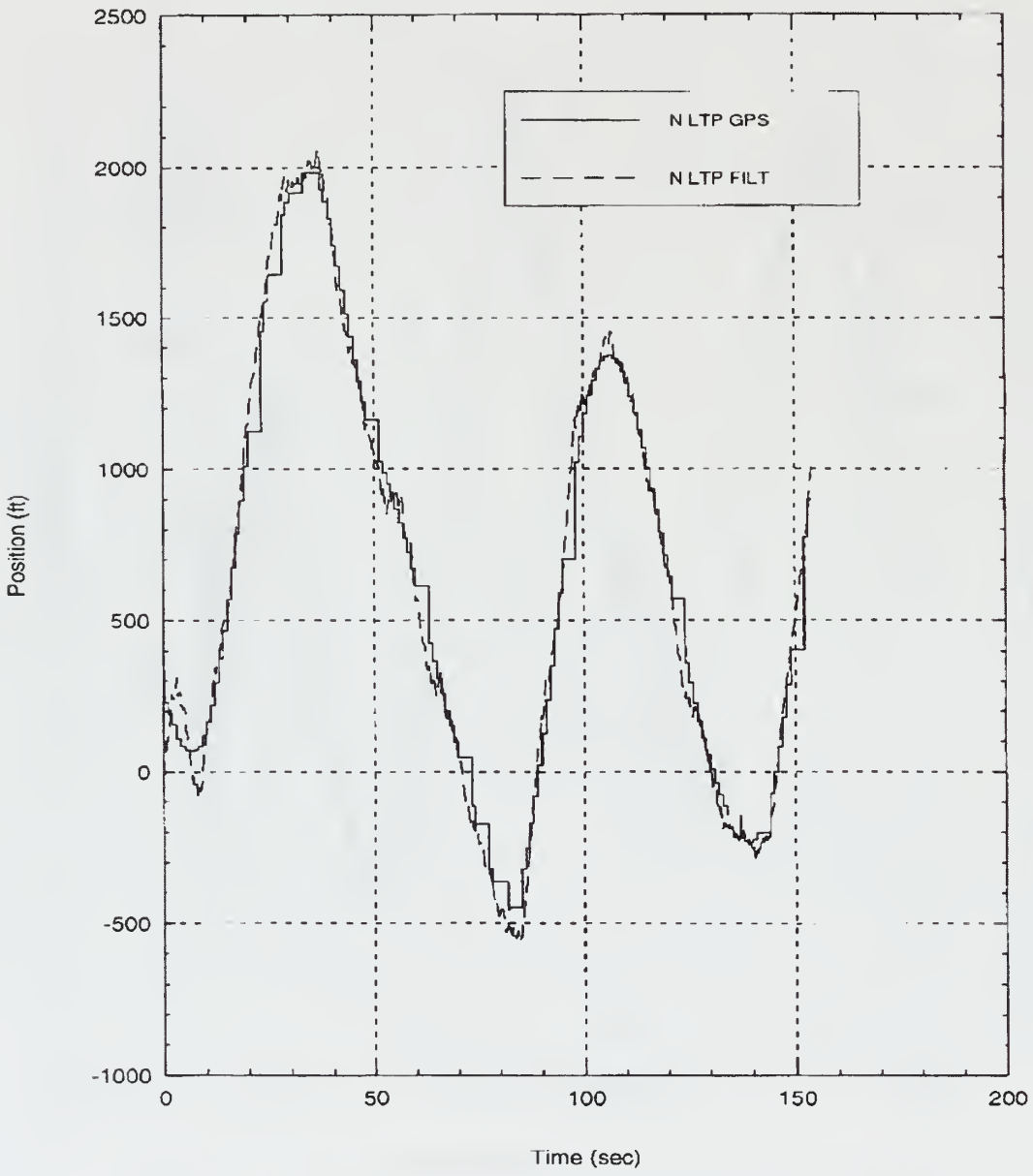


Figure 6.10: North DGPS and Filter Estimated Position Using Psi Model

The magnitude of the estimated position error in the absence of DGPS update is reduced by approximately 70 percent. This is accounted for in the availability of accurate

25 Hz heading information from the model. From this it is clear that quality IMU heading information would greatly increase the accuracy of the filter position estimate.

Down channel LTP position estimate in Figure 6.11 is, as expected, the most oscillatory due to the lack of theta and phi estimates. Since there is very little if any wind component in the down channel, the filter relies on the DGPS updates as the sole means of determining position error.

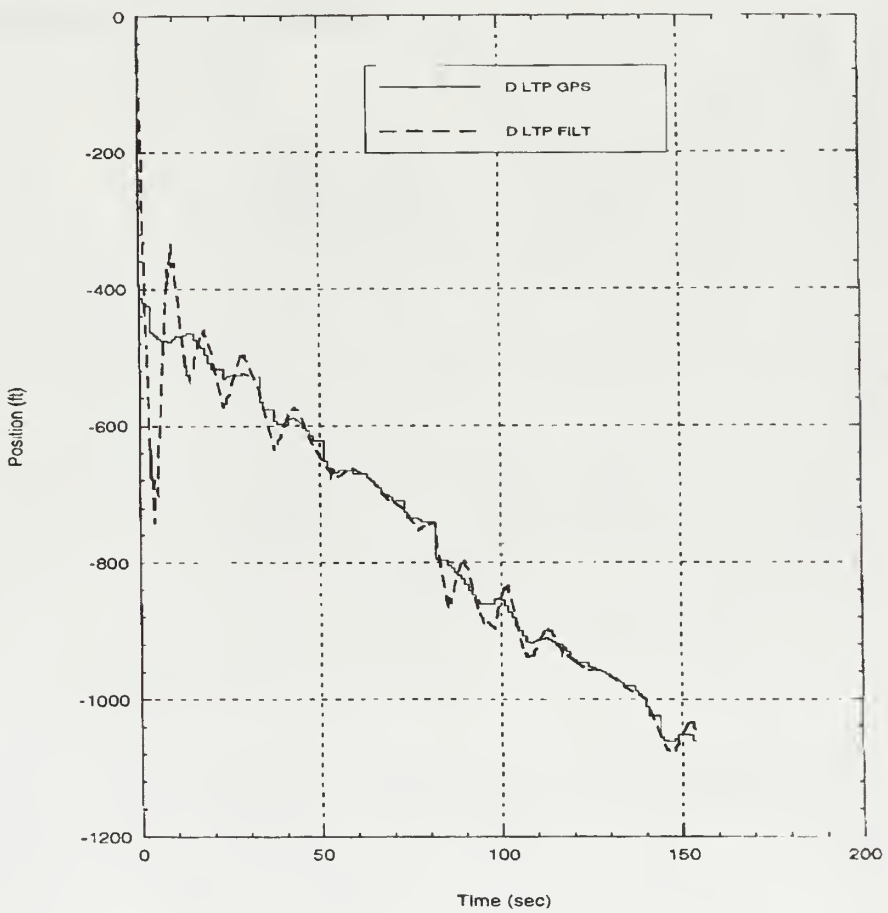


Figure 6.11: Down DGPS and Filter Estimated Position

4. Wind Estimate

Accurate position estimation is a function of the wind estimate made by the filter. As discussed in Chapter III the filter should accurately estimate the existing steady state wind while rejecting the gusts. Figure 6.12 shows the unfiltered wind estimate output in all three channels.

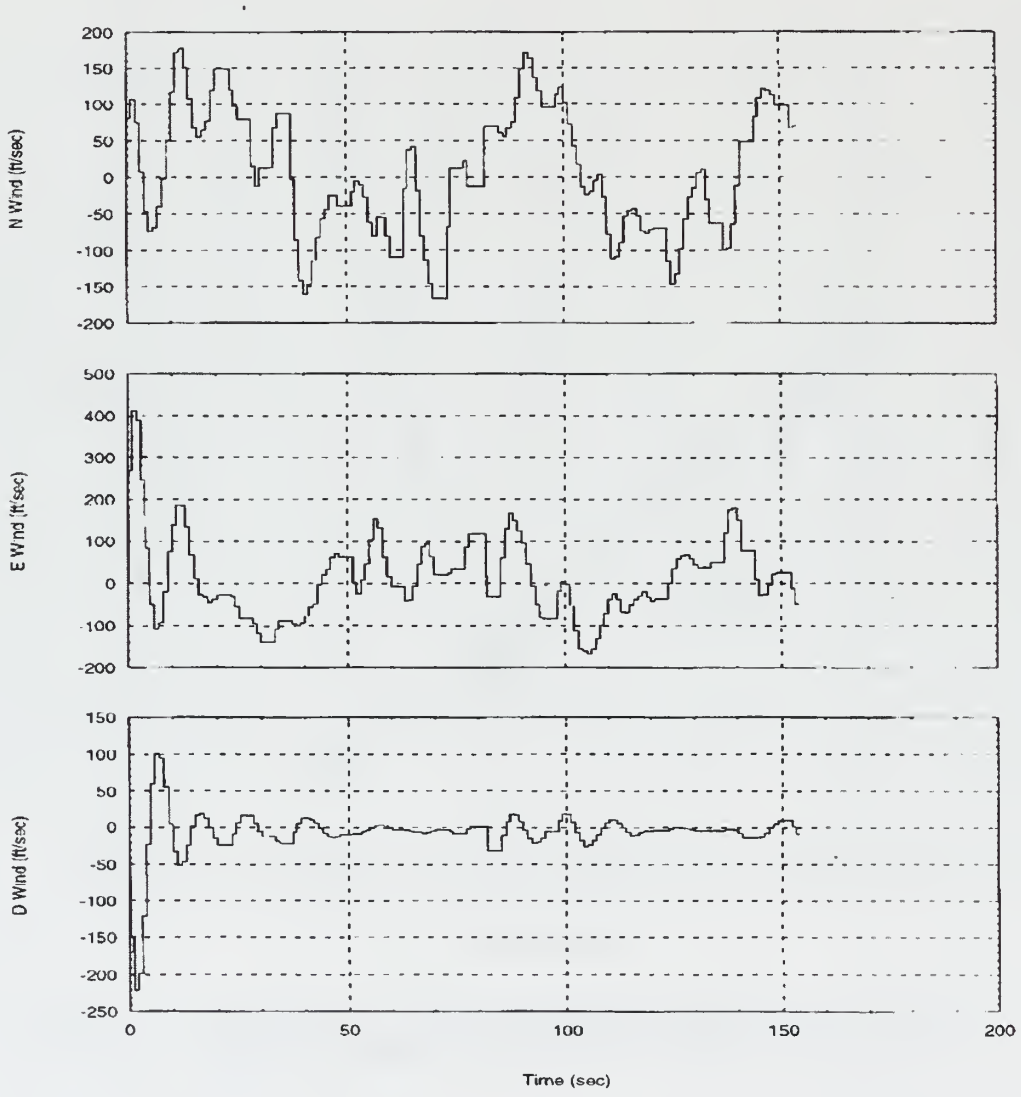


Figure 6.12: North, East and Down Filter Wind Estimate

It is clear from Figure 6.12 that this wind information contains high frequency components. By inspection the down channel wind is estimated to be negligible even without filtering. Such is not the case with the north and east components. In order to extract the desired information, the prevailing wind condition, it is necessary to filter the north and east signals. To this end a Butterworth filter with a cutoff frequency of .005 Hz was employed after the wind estimate integrator and the resulting low-pass filtered wind signals displayed on the IA LTP Navigation page. A sample of the filtered wind results from flight test are shown in Figure 6.13.

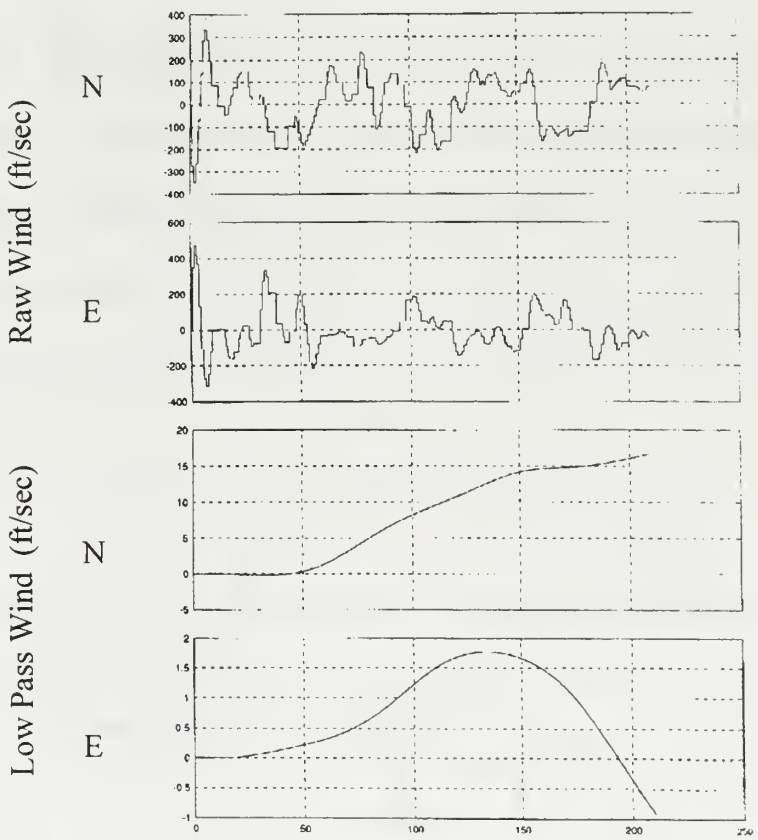


Figure 6.13: Raw and Filtered Wind

Noticing scaling, these results show the major wind component in the N channel with a magnitude approaching 15 to 20 fps. This approximates the actual north wind of 15 to 25 fps in which that flight test data was taken. The long delay in the filtered wind is caused by the Butterworth filtering. Longer data set would refine the estimate.

5. Bank Angle Estimate

Although the bank angle information provided by the IMU in the form of Euler angle phi was unusable, it was possible to generate and estimate of phi using the relationship:

$$\phi = V \tan^{-1} \frac{\dot{\psi}}{g}. \qquad 13$$

Heading rate needed for this calculation was available as an output from the **simple_complementary_filter_psi** SuperBlock. This signal was filtered using a Butterworth filter with a cutoff frequency of .05 Hz. Airmass velocity resolved in inertial frame was input from the position filter. This calculation was carried out in the **phi_filt** SuperBlock located within the **Sensor Filters** SuperBlock. The implementation is shown in Figure 6.14.

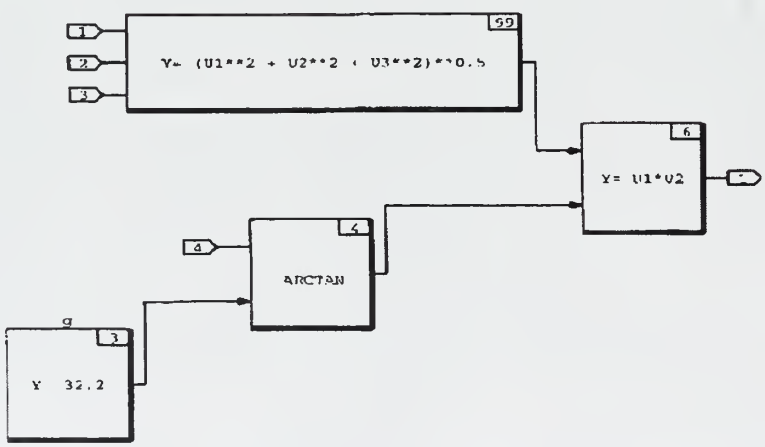


Figure 6.14: Phi Filter SuperBlock

A sample of the bank angle estimate generated using the phi_filt implementation with the FROG in a constant turn is shown in Figure 6.15.

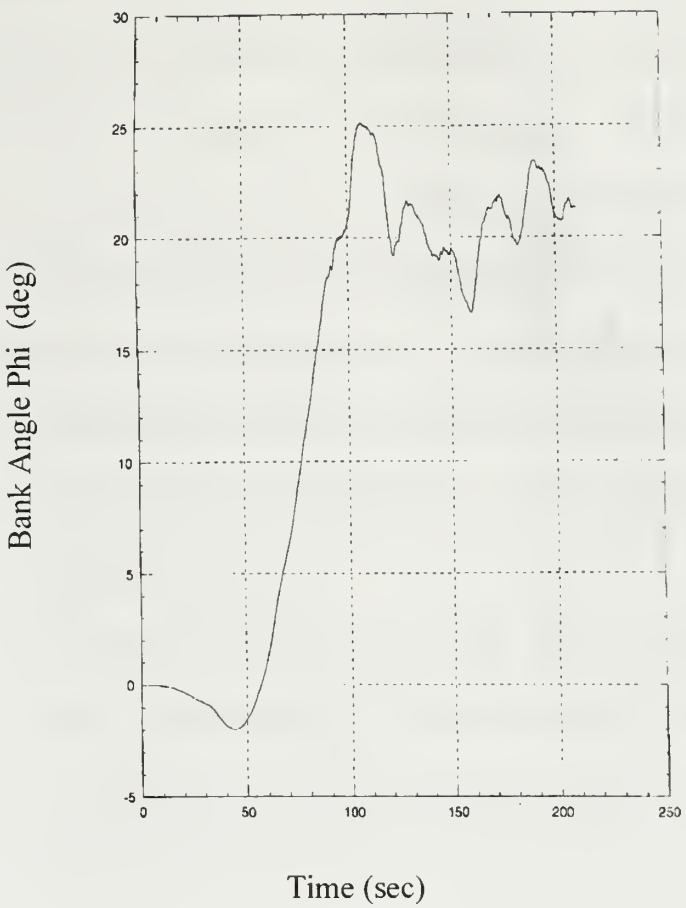


Figure 6.15: Phi Estimate Using Heading Rate

A comparison of this phi estimate with the actual flight profile for this data set shows the estimated results to be close to the angle of bank used in the steady turn for flight test. As with the wind filter, the large delay is caused by the use of the Butterworth filter. Although this signal would not be suitable for control purposes due to the lag, an improved yaw rate source would alleviate the need for the Butterworth filter for $\dot{\psi}$.

These results are very close to those obtained through simulation under similar flight conditions. A sample of the simulated phi estimate is shown in Figure 6.16.

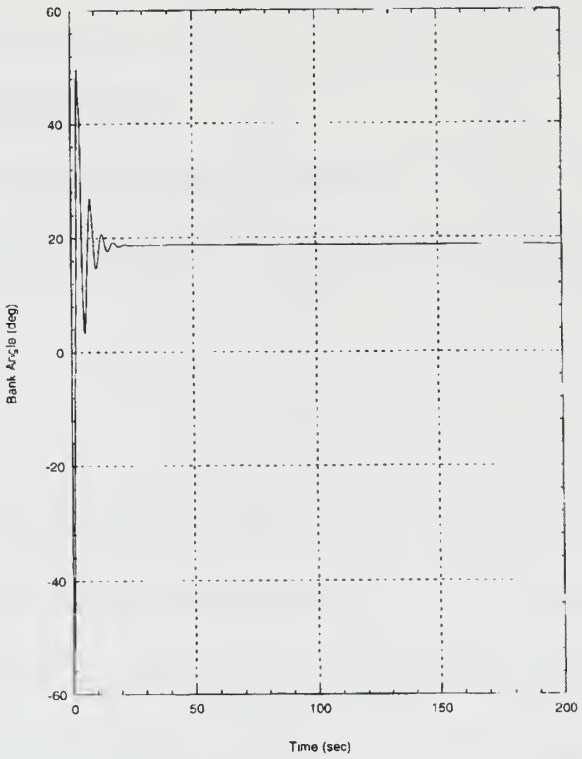


Figure 6.16: Phi Estimate from Simulation

VII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The primary goal of this thesis was to design a multi-rate position filter to provide accurate position estimates between the 1 Hz DGPS updates. This was accomplished using complementary filtering of the DGPS position and of indicated airspeed both resolved in the LTP inertial frame. In the implementation of this filter the heading information necessary in the transformation from {u} to {b} was provided by the DGPS. Filter performance in simulation and flight test revealed the ability of the filter to estimate the wind and provide accurate position information was mainly a function of availability of DGPS position information.

Analysis of the flight test performance of the filter displayed its ability to accurately estimate vehicle position despite poor DGPS update rates. Further filtering of filter generated wind provided the user with an accurate low frequency estimate of the existing wind conditions.

Despite successful demonstration of this filter the two problems of poor DGPS update rates and slow IMU heading information yet to be solved.

B. RECOMMENDATIONS

1. DGPS Update Rate

As evidenced in the analysis of the flight test data filter performance is a function of the DGPS update rate. Latency in the receipt of DGPS position information disables the evaluation of the existing wind by the filter resulting in poor position estimates. An examination of the data sets from the past eight flight tests shows DGPS update rates

averaging between .73 and .78 Hz well below the advertised 1 Hz nominal. The cause of this update problem has been isolated to the Freewave RF modems used to provide the communication link between the ground station and the UAV. An adjustment of the communication protocols must be made in the software that would allow the proper establishment of two-way communication between the ground and UAV units.

2. Heading Information

Flight test results revealed the poor quality of nearly all IMU data. Consequently only the roll, pitch and yaw rates from the IMU were ultimately used in the current version of the filter. IMU heading information was noticeably slow, appearing to have been filtered. IMU settings must be analyzed to determine if the current data is being sent in filtered or raw form. If it is found to be raw data then the IMU must be removed and replaced with a unit that as a minimum will provide adequate heading information for standard rate turns of 12 deg/sec. Should the examination of the IMU settings reveal the present data is being sent in a filtered form, the settings should be changed to send raw heading information.

LIST OF REFERENCES

1. Allen, P. M., *Incorporatation of a Differential Global Positioning System (DGPS) in the Control of and Unmanned Aerial Vehicle (UAV) for Precise Navigation in the Local Tangent Plane (LTP),* Master's Thesis, Naval Postgraduate School, Monterey, CA March 1997.

2. Marquis C., *Integration of Differential GPS and Inertial Navigation using Complementary Kalman Filter,* Master's Thesis, Naval Postgraduate School, Monterey, CA, 1993.

3. Kaminer, I, I., *AA3276: Intro to Avionics,* Course Notes, Naval Postgraduate School, Monterey, CA, Sep 1996.

4. Schmidt, L. V., *Introduction to Aircraft Flight Dynamics,* Draft, Naval Postgraduate School, Monterey, CA, November 1996.

5. Zanino, J. A., *Uniform System for the Rapid Prototyping and Testing of Controllers for Unmanned Aerial Vehicles,* Master's Thesis, Naval Postgraduate School, Monterey, CA, September 1996.

6. Integrated Systems Inc., *$MATRIX_x$ Product Family System Manuals,* Version 5.0, 1996.

7. Papageorgiou, E. C., *Development of a Dynamic Model for a UAV,* Master's Thesis, Naval Postgraduate School, Monterey, CA, March 1997.

8. Hallberg, E. N., *Design of a GPS Aided Guidance, Navigation, and Control System for Trajectory of an Air Vehicle,* Master's Thesis, Naval Postgraduate School, Monterey, CA, March 1994.

9. Kaminer, I. I., *Class Notes for AA4342,* Naval Postgraduate School, Monterey, CA, 1997.

10. Kaminer, I. I., *AA4276 Lecture Notes,* Naval Postgraduate School, Monterey, CA, 1996.

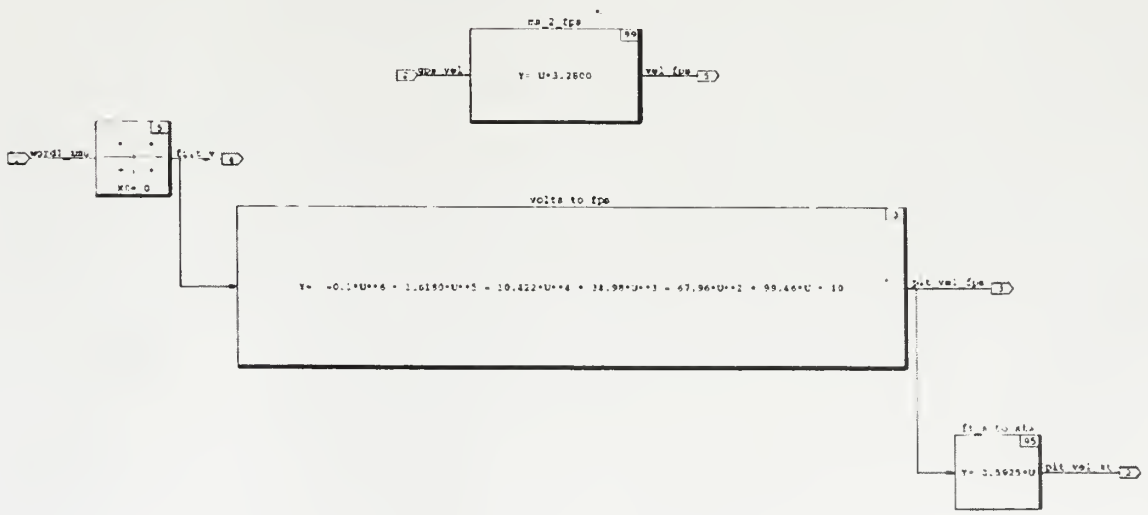
11. Moats M. L., *Automation of Hardware-in-the-Loop Testing of Control Systems for Unmanned Air Vehicles,* Master's Thesis, Naval Postgraduate School, Monterey, CA, 1994.

APPENDIX A. INDICATED AIRSPEED CALIBRATION

Indicated airspeed was sensed by the pitot-static system on the FROG UAV. This measured voltage was discretized using the IMU Analog to Digital Converter (ADC) and then transmitted to the ground station via the Freewave RF modem. The discretized voltage sensed by the ground station was quite noisy and consequently put through a simple smoothing filter. Using data from numerous flight tests, the mapping from sensed voltage to indicated airspeed in units of feet-per-second was established [Ref. 7].

$$Y = -0.1 \cdot U^6 + 1.618 \cdot U^5 - 10.422 \cdot U^4 + 34.98 \cdot U^3 - 67.96 \cdot U^2 + 99.46 \cdot U + 10$$

This conversion which takes voltage as the input was again modified following comparison to DGPS velocity under no wind conditions. Below is the airspeed conversion SuperBlock found in the Calibrate Sensors SuperBlock of the flight_test_rf.rtf file.



Airspeed Conversion SuperBlock

APPENDIX B. DGPS UPDATE LOGIC BLOCKSCRIPT

This C-code is found in the Plogic SuperBlock within the position filter. This simple logic compares the norm of the current DGPS position signal with that of the previous. If the difference of that operation is zero, there has been no update and the output of the logic is low (zero). If the difference is non-zero, there has been and update and the logic output is high (one). This output triggers the data path switches which turn the feedback loop gains on and off.

```
Inputs: u;
Outputs: y;
States: x;
next_states: xnext;
float u(:), y, x;
x=(u(1)^2 + u(2)^2 + u(3)^2)^0.5;
if x ==  0 then
        y = 0;
else
        y=1;
endif;
```

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library.....2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Dr. Isaac I. Kaminer, Code AA/KA.....3
   Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, California 93943-5121

4. Dr. Russell W. Duren, Code AA/DR.....1
   Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, California 93943-5121

5. Department of Aeronautics and Astronautics.....1
   Code AA
   Naval Postgraduate School
   699 Dyer Rd. Rm. 137
   Monterey, California 93943-5106

6. Lieutenant Commander Robert C. Perry.....3
   21002 Woodmere Rd.
   Leonardtown, Maryland 20650